

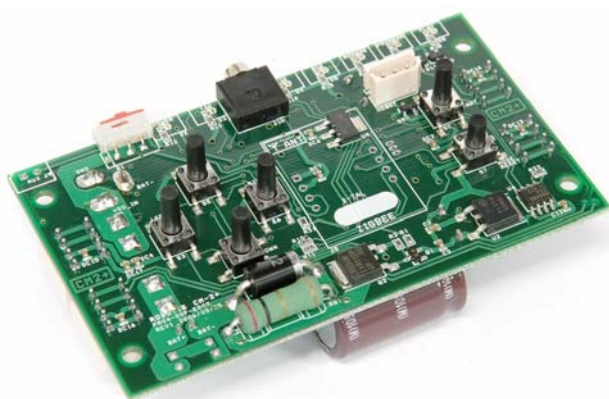
Ver 1.1

Closer to Real

**ROBOTIS**

Dynamixel Dedicated Controller

CM-2+



## Table of Contents

<b>1. Before Starting</b>	
1-1. A Word of Caution	Page 4
1-2. What is the CM-2+ ?	Page 5
1-3. Things to Understand Before Starting	Page 6
1-4. CM-2+ and Optional Items	Page 10
<b>2. Learning the Basic Operations</b>	
2-1. The CM-2+ and Its Operation Modes	Page 11
2-2. Connecting the Frames	Page 13
2-3. Wiring	Page 14
2-4. Understanding ID, Address, and Data	Page 15
<b>3. Behavior Control Program</b>	
3-1. Starting Behavior Control Program	Page 18
3-2. File	Page 20
3-3. Grammar of Behavior Control Program	Page 21
3-4. Edit	Page 29
3-5. Running	Page 32
3-6. Using CM-2+	Page 34
3-7. Using Dynamixel	Page 38
3-8. Using Sensors	Page 46
3-9. Error Code	Page 58
<b>4. Motion Programing</b>	Page 77
4-1. Starting Mortion Editor	Page 61
4-2. Creating Motions	Page 64
4-3. Playing Motion	Page 71
<b>5. Robot Terminal</b>	Page 77
<b>6. Wireless Remote Control</b>	
6-1. Infrared Wireless Maipulation	Page 85
6-2. ZIGBEE Wireless Manipulation	Page 89

**7. Management Mode**

7-1. Setting the ID and Dynamixel Search	Page 92
7-2. Other Commands	Page 97

**8. Information for Advanced Users**

8-1. Boot Loader	Page 103
8-2. Using the C Program Language	Page 106
8-3. Compiling	Page 107
8-4. Example.c	Page 115
8-5. Circuit Diagram	Page 125
8-6. Understanding CPU Port	Page 128

**9. CM-2+ Program Update**

9-1. CM-2+ Program Update	Page 130
9-2. Dynamixel Program Update	Page 133

**APPENDIX**

CM-2+ Board Diagram	Page 136
---------------------	----------

# 1. Before Starting

## 1-1. A Word of Caution

### A Note on Safety

You are responsible for any accidents that occur while wiring and setting up the product. Before starting, please remember the following.

- Read and study the manual before starting.
- The product is not waterproof so be cautious when handling near water.
- Do not operate or store it in under direct sunlight.
- Do not operate or store it near open flames or in humid environments.

### Board Malfunction

If any of the following occurs, immediately turn off the power and contact an adult supervisor or the company.

- When you see smoke coming out of the product.
- When the LED does not turn on after power is connected.
- When water or foreign substances enter the robot.
- When you detect an unnatural smell from the product.
- When the robot is damaged.

### Notes

Please note the following.

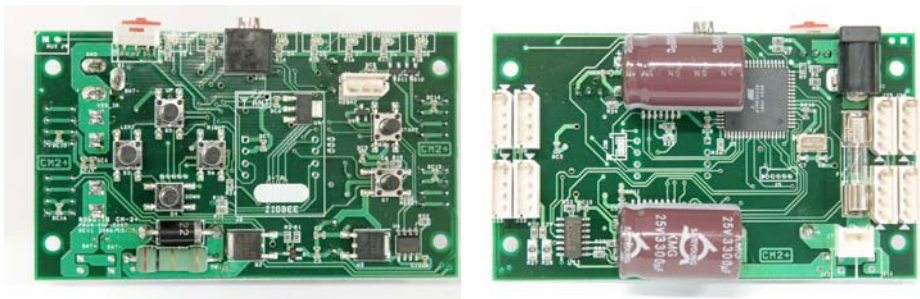
- Before operating a Dynamixel cable, check the pin order of the connector.
- Turn off the power immediately to avoid damage to the robot if a joint gets twisted from wild motions during development.
- When operating the robot with the SMPS, make sure the robot does not fall and refrain from wild movements. This can cause the SMPS cable to break.
- Before connecting a battery, check the pin order of the connector that connects to the battery.



## 1-2. What is the CM-2+ ??

### CM-2+

The CM-2+ is a central control device that controls the Dynamixel series. Through the CM-2+, you can operate the Dynamixel motors and read information from the sensor. The CM-2+ also allows you to construct and program joint type robots.



[CM-2+ ]

### Function

The Dynamixel motor series, DX-113, DX-117, RX-28, and RX-64 link to the AX-S1, a Dynamixel sensor, allowing you to edit the motion of robot, create a behavior control program, and program in C language.

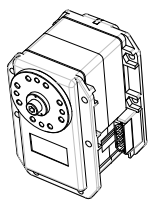
## 1-3. Things to Understand Before Starting

Before using the CM-2+ , there are a few basics that you have to understand. First, let's learn about the robot's hardware and software. The terms used here will be mentioned throughout the manual so it is important that you understand them.

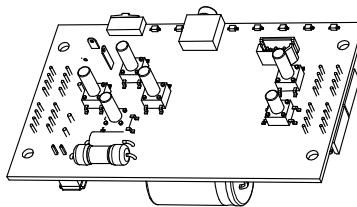
### Hardware

The hardware of the robot that uses CM-2+ consists of three types.

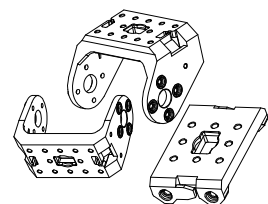
- **Dynamixel:** This is the basic unit of the robot that uses the CM-2+ , which can be either a joint or a sensor. The DX-113, DX-117, RX-28, and RX-64 are actuators used as joints. The AX-S1 Dynamixel is a sensor unit that can sense both distance and sound.
- **CM-2+ :** This is the control board and the brain of the robot.
- **Frame:** The frame connects the robot units. The Dynamixel units can be connected together using the frame. The frame also connects the Dynamixel units and the CM-2+ unit.



[Dynamixel]



[CM-2+ ]



[Frame]

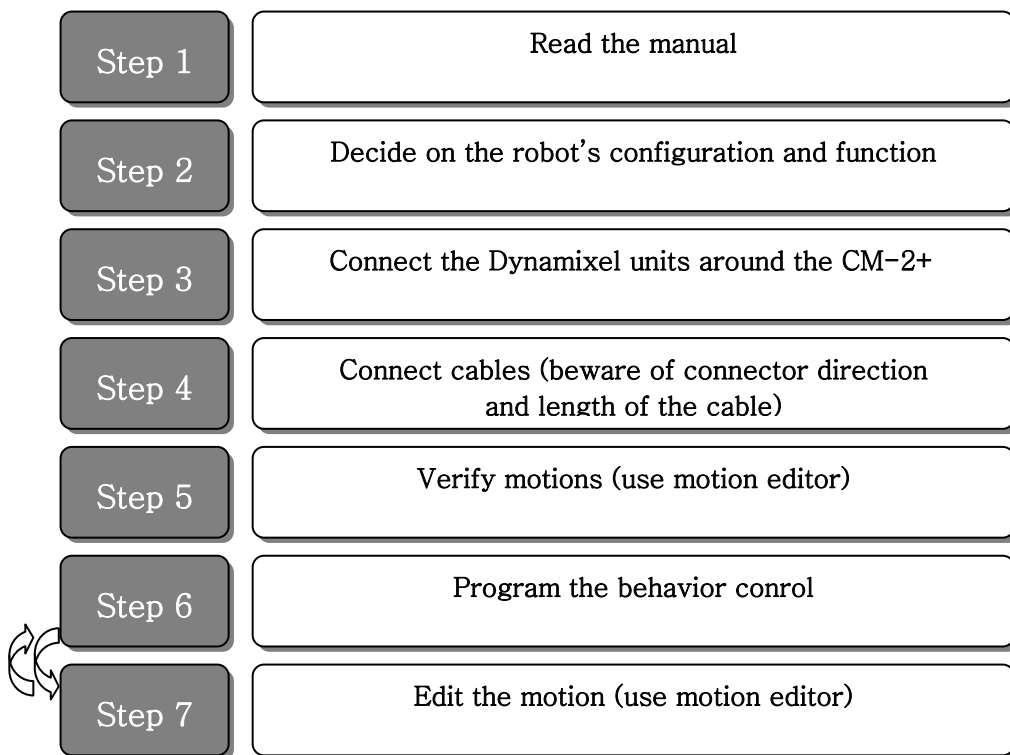
### Software

There are three pieces of software supplied with the CM-2+ .

- **Behavior control programmer:** This is used for creating a program that controls the robot's behavior. The program is used to implement a motion the robot follows according to the information received through input devices such as sensors.
- **Motion editor:** The motion editor is software to help create robot motions and recall them as necessary.
- **Robot terminal:** This is a serial communication terminal program that advanced users often use to see information displayed on the screen sent by the robot and to send the characters typed on the keyboard to the robot.

## Assembling Process

The steps to develop a robot that uses the CM-2+ are as follows.



- Step 1** Read and completely understand the manual before trying to build a robot. The manual consists of nine chapters with the first five chapters intended for beginners.
- Step 2** This is where you decide what kind of robot to build. There are many different kinds of robots that use the CM-2+.
- Step 3** Step 3 is the building stage. First, connect the Dynamixel units to the CM-2+ unit as the center unit. Connect other Dynamixel units to this to create joints and expand to complete the robot. Secure each part with nuts and bolts.

**Step 4**

After assembling the robot, the next step is to connect the cables. Using the CM-2+ unit as the center, connect the wires to the Dynamixel units in a daisy chain fashion. Each cable is made up of three wires [AX-S1] or four wires [Dynamixel other than AX-S1] with two of the wires for power and one for communication. Make sure the cables are long enough to allow the joints full range of motion. Take note that for the AX-S1, use three wires and connect it to the 3P connector (J4) of the front of the CM-2+ and for DX-113, DX-117, RX-28, and RX-64, use the back of the 4P connector (J30~J37) to connect.



The steps up to here complete the building of the robot hardware. The configuration of the robot was decided in Steps 1, 2, 3, and 4 and the function of the robot will be decided through programming in Steps 5, 6, and 7.

**Step 5**

After you are done building the robot, use the motion editor program to make sure the robot is properly assembled. Make sure all the Dynamixel units communicate with the CM-2+ unit properly. To check if the joints are working properly, test them by slightly moving each of the joints.

**Step 6**

In step 6, you will create the behavior control program of the robot. Behavior control is simply telling the robot to take some kind of action when it enters a certain state. For example, when the robot is walking forward through a narrow path, you can make it walk through the center of a path or, if there is something blocking the way, you can make it turn around and go the other way. The behavior control program takes the form of rules which define the appropriate motion the robot should use for specific input information it receives from the sensors.

**Step 7**

In step 7, you will create the motions of the robot. For robots that use wheels, step 7 is unnecessary because all you have to do to move the wheels is change the position or velocity settings of the Dynamixel units. It would be difficult to make a dog robot sit or a humanoid robot walk by changing each of its joint angles individually. In order to move a complicated multi-jointed robot, you have to “play” a “pre-recorded motion.” These “pre-recorded motions” are what you will create here in step 7. The behavior control program of step 6 will be able to “play” these “pre-recorded motions” of step 7. Complete step 6 before moving on to step 7.

**PC Requirements**

- PC : IBM compatible (Required)
- OS : Windows 2000 or Windows XP (Required)
- CPU: Intel Pentium III 1GHz or AMD Athlon XP 1GHz or higher (Recommended)
- RAM: 256MB or higher (Recommended)
- Graphic Card : 3D acceleration function (Direct 3D supported) (Required)
- HDD free space : at least 300MBytes (Recommended)
- Direct X 8.0 or higher (Required)

## 1-4. CM-2+ and Optional Items



CM-2+



Serial cable

Included in the box



SMPS (12V, 5A)

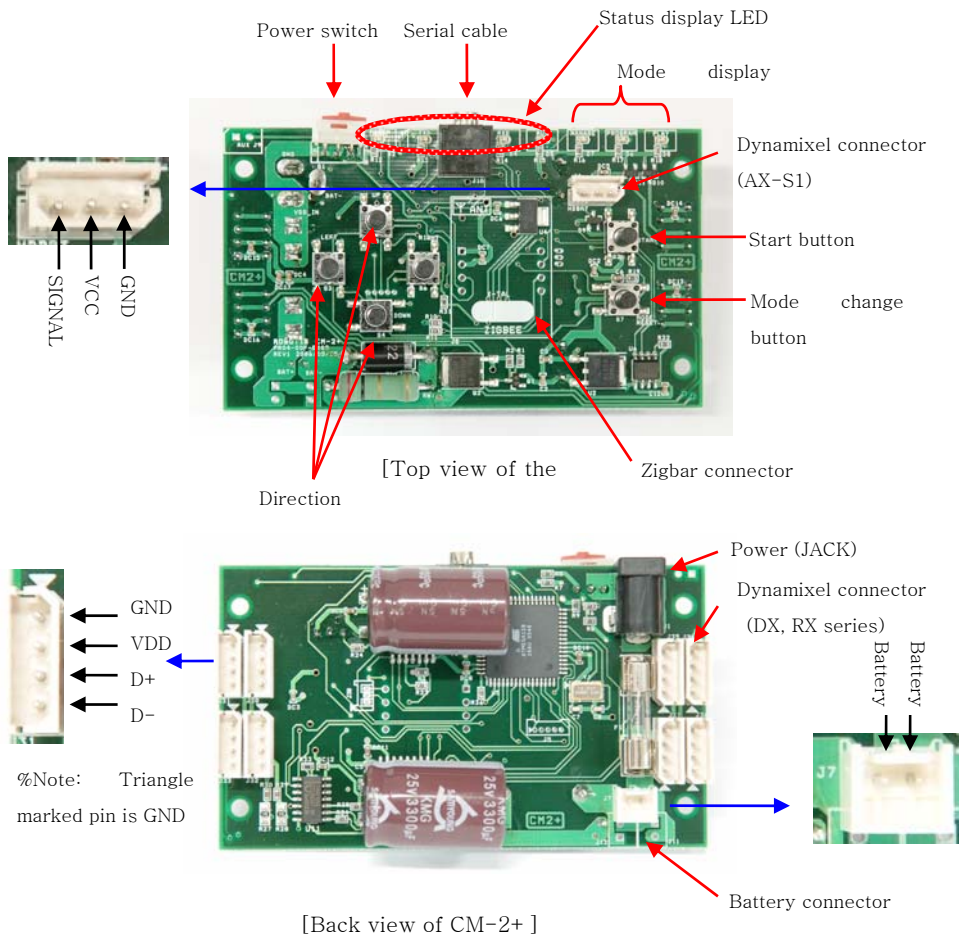
Optional items

## 2. Learning the Basic Operations

### 2-1. The CM-2+ and its Operation Modes

#### CM-2+

The CM-2+ is the central control unit and power source for the robot. As mentioned earlier, the robot is built by connecting the Dynamixels to the CM-2+ as the central unit. In order to understand how the robot works, you first have to understand how the CM-2+ operates.



## Applying Power

Apply power to the CM-2+ unit by plugging the SMPS into the power jack on the upper left corner. Then turn the power switch on. One of the mode display LEDs should be blinking. As you press the mode change button, the mode LED will change sequentially. Initially, it is in standby mode. The recommended specification of the SMPS is 2V~18V, 5A.

**Operating Modes** -The operating modes of the CM-2+ unit is as follows

- **Manage mode:** This is used when you want to know the status of the CM-2+ unit or the Dynamixel units, or when you want to test motions. This mode should only be used by advanced users who are very confident with operating the robot.
- **Program mode:** The mode used for editing the motion.
- **Play mode:** The mode used for running the behavior control program created.
- **Standby mode:** The mode before running the other three modes.

## Execution

If you press the start button during standby mode, the CM-2+ unit will go into the selected mode. To go back into standby mode, press the mode change button or turn the power switch off and then back on.

### TIP

The mode change button is the reset button for the CPU inside the CM-2+ unit. Therefore, when the power is on, the CM-2+ will go back to standby mode whenever the mode change button is pressed.

## Serial Cable

In order to communicate with a PC, the CM-2+ unit has to be connected using a serial cable.

- **Concerning laptops:** Since most laptops do not have a serial port, you have to use a USB to serial converter device. A USB2Serial converter can be purchased at local computer stores.

## Status Display LED

Four LEDs indicate the status of the CM-2+ unit. The definitions of each are as follows.

- **Power:** If the power is on, this LED will be on.
- **TXD:** This LED is on when the CM-2+ unit is transmitting data.
- **RXD:** This LED is on when the CM-2+ unit is receiving data.
- **AUX:** This LED is assigned for user programming. It can be turned on or off with the behavior control program.

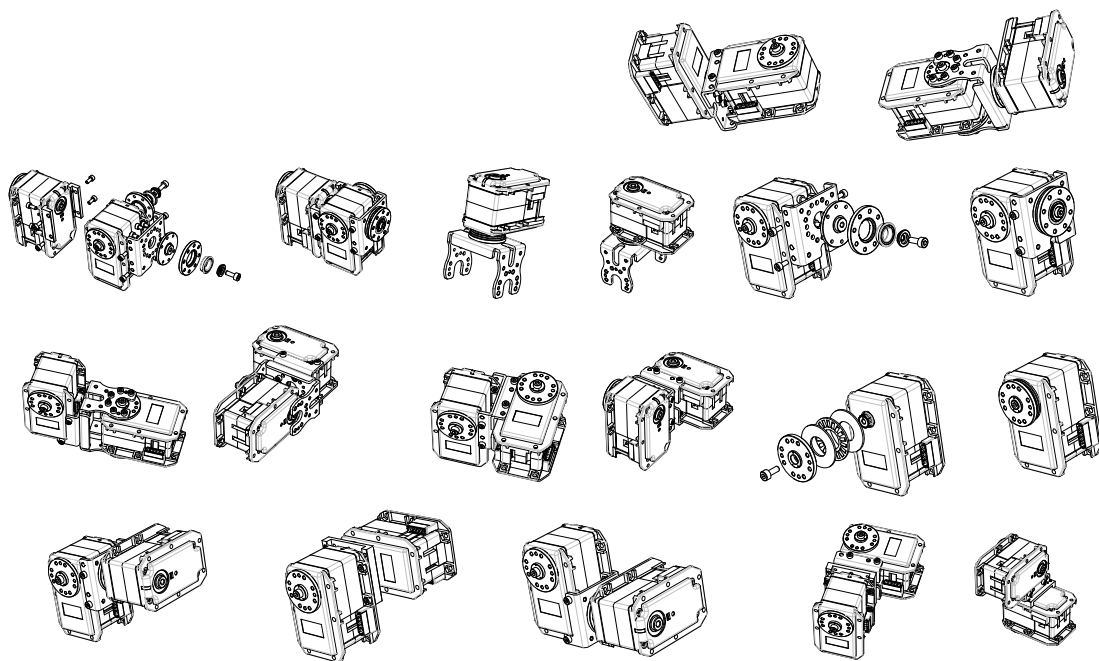
## Direction Buttons

These buttons are also assigned for user programming similar to the AUX LED.

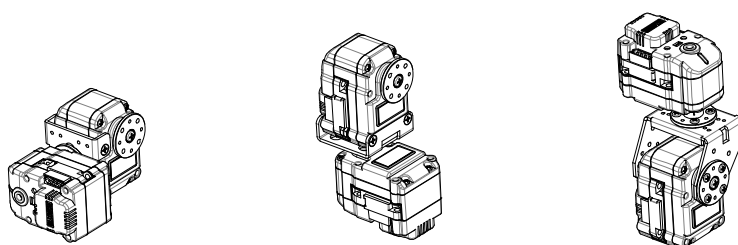


## 2-2. Connecting the Frames

Refer to the applicable manual and assemble optional frame and created frame with Dynamixel. The following is an example using the optional frames.



[RX-64 and optional frames]

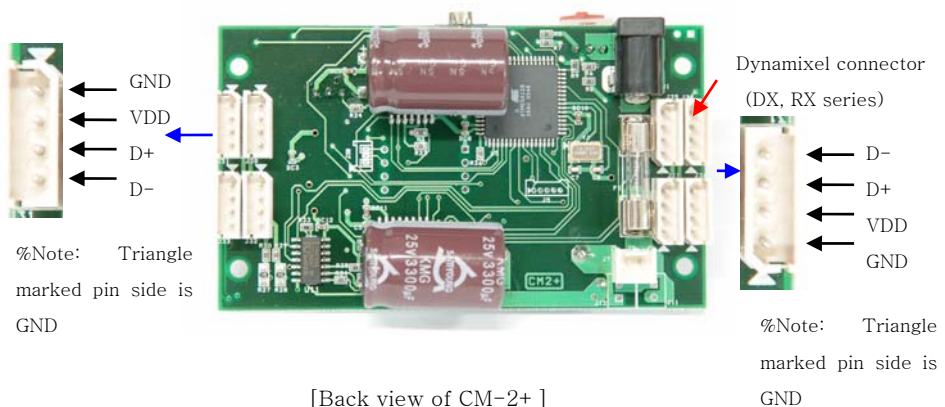
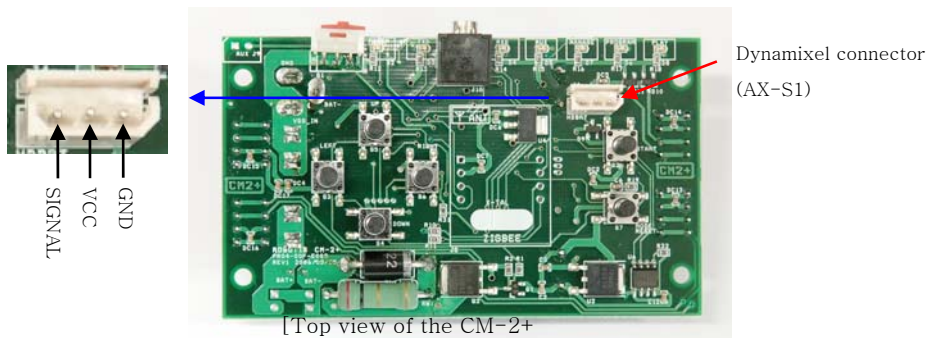


[DX-113, DX-117 and optional frames]

## 2-3. Wiring

Wiring is an important part of building the robot. Many problems can occur with the robot due to faulty wiring. Problems that may occur include:

1. Severed cables that were too short
2. Cables getting stuck in between the robot's (mostly occurs when users want to assemble the robot in an uncluttered fashion and wire the cable through narrow joint spaces.)
3. Self-made cables: Take extra precaution with the pin configuration of the connector for self-made cables.



As shown in the figure above, the four pin connection (RS-485) of the Dynamixel motors (DX-113, DX-117, RX-28, RX-64), are connectors J30~J37 located on the back of the CM-2+, whereas, the three pin connection (TTL level) of the Dynamixel sensor (AX-S1), is located on the front of the CM-2+.

The cables should be connected in a daisy chain configuration, as shown in the figure below.

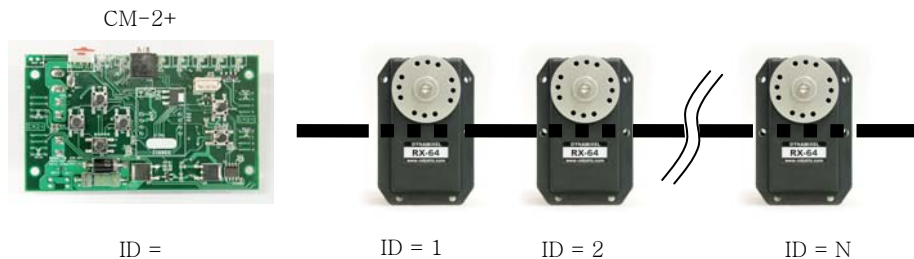


## 2-4. Understanding ID, Address, and Data

So far we have learned how to create a simple behavior control program. Now let's learn about the available input and output items.

### ID

As mentioned earlier, the robot is made up of Dynamixel units and a CM-2+ unit. The input and output items all exist inside these devices. All of the robot modules are connected to one bus and each module has its own unique ID.



### Address

Each Dynamixel unit has many input and output items. Each item is numbered with an address.

### Data

Data is the value of each input and output item. If you look at the behavior control program created earlier (ButtonAndLED.bpg), you can see that the LED is located in the LED item (Address 24) of the CM-2+ (unit with ID = 200). We made an "If" statement to see if the data it contains is 8 (the data value for the up button). Let's run the behavior control program again and look at the "If" statement with this in mind.

Setting the input and output items requires specifying the ID and address.

The robots using the CM-2+ have three different units: the CM-2+, Dynamixel motors (DX-113, DX-117, RX-28, RX-64) and sensors, the AX-S1. Let take a look at the available input and output items.

**ID Assignment** The following are IDs that can be assigned to each unit.

- Dynamixel motors: ID = 1 ~ 19 (allowed range is 0~30)
- Dynamixel motors [AX-S1]: ID = 100 (allowed range is 100~109)
- CM-2+: ID = 200

For the Dynamixel units, you can set or reset the ID as needed. The Dynamixel units that come directly from the factory have their IDs set to 1. If you have a problem with one of these units and have to order a new one, the IDs need to be reset accordingly. To reset the ID, refer to the "Maintenance Mode" section, or 3-3. Additionally, you can refer to video clips for changing a Dynamixel's ID in the directory "helpWID changing.wmv."



### 3. Behavior Control Program

A robot can behave in many different ways. However, it can do so only if there is a program that tells how the robot should act for certain situations. This program is called the “behavior control program.” A behavior control program is a series of rules that define what actions a robot should take for given states. Insert the provided CD into the PC and install the software used for creating the behavior control program.

**Installation**      **Install the following three programs.**

- Behavior Control Programmer
- Motion Editor
- Robot Terminal

#### **Input and Output**

The behavior control program takes the form of a series of rules that mutually connect the input and output. For example, let's say that we want to build a robot dog that stands up when you clap once and sits down when you clap twice. The input item (clap once or twice) and the output item (stand up or sit down) have to be predefined. Behavior rules need to be defined that tell what behavior to output for the given input item (clap once or twice).

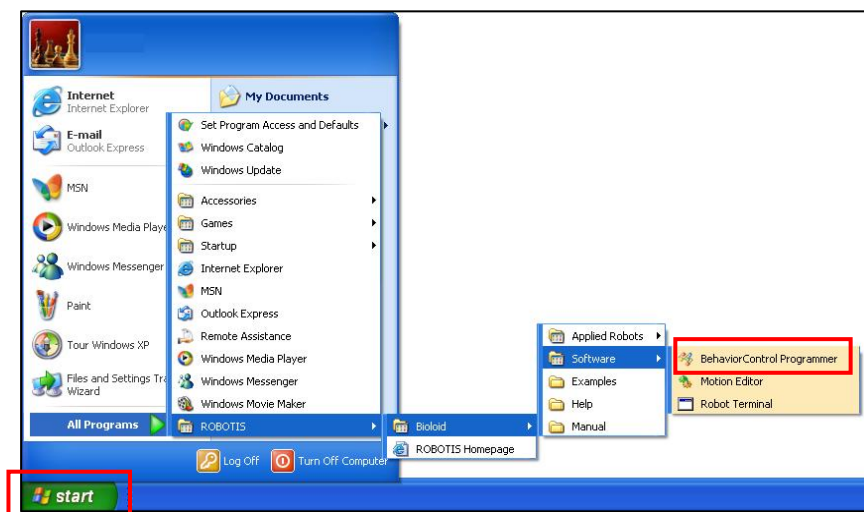
The following are the general expression sentences for these behaviors.

- If input item 1 occurs, then execute output A.
- If input item 2 occurs, then execute output B.

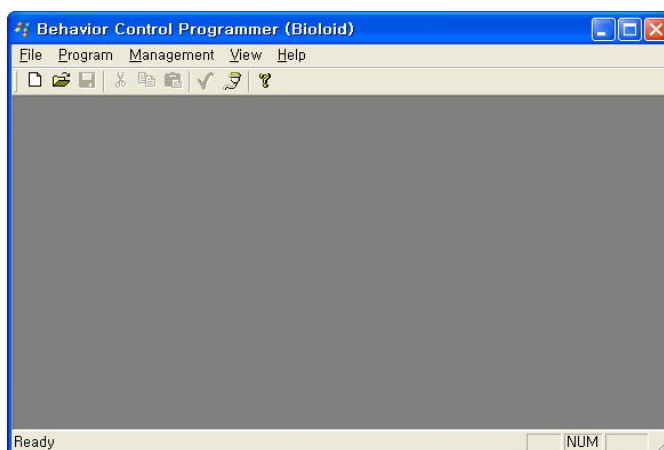
Thus, understanding what types of input and output items are available is important when writing a behavior control program. Part of learning how to use the robot is knowing the input and output items. Let's practice creating a simple behavior control program using the CM-2+ unit's input and output items.

### 3-1. Starting Behavior Control Program

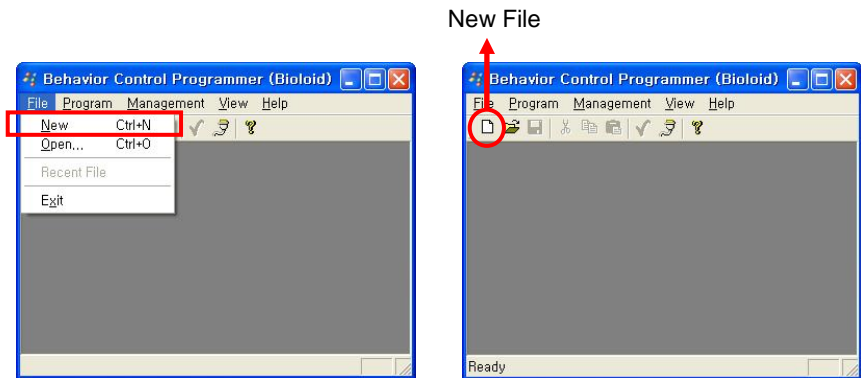
Select 'Start >> All the programs >> Robotis >> Bioloid >> Software >> Behavior Control Programmer' on the window as in the following figure then the behavior control programmer starts to run.



Initial page of behavior control programmer



To go to the behavior control program, select 'File >> New File' as in the left figure below or simply click the shortcut key corresponding to 'New File' as shown in right figure below.



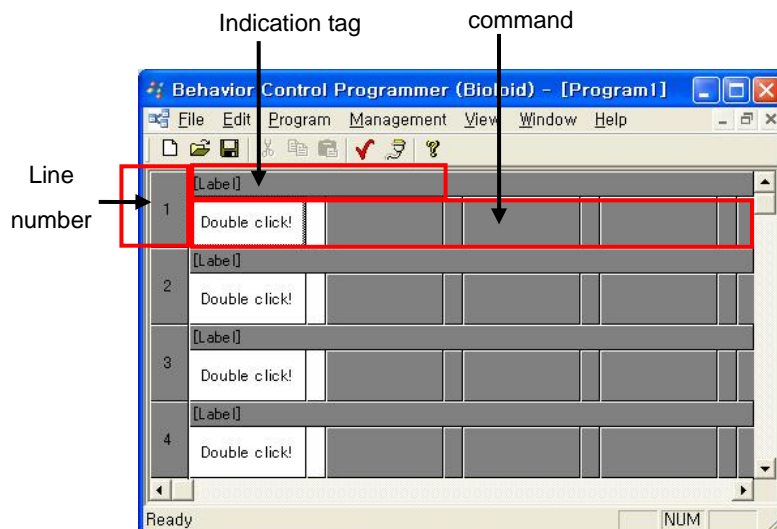
The behavior control program consists of multiple lines.

Each line consists of several items; line numbers, indication tags, and commands

Line numbers are used to edit each line such as selecting, copying, deleting.

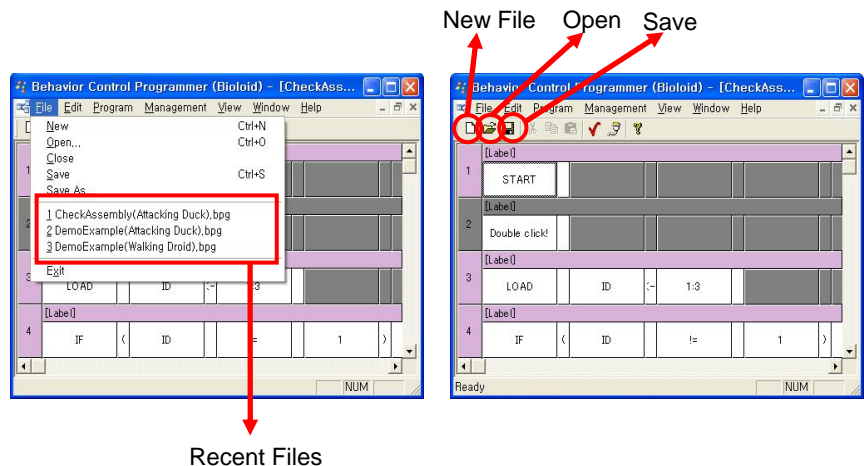
Indication tags are names of lines. They are used for performing functions such as call or jump.

Spaces of commands are used to make directions. Directions consist of numbers of blocks. Double-clicking a block shows a proper command.



### 3-2. File

When you select 'File' on the behavior control programmer's menu, you can see the following items. Since these items have functions similar to those of the general text editing program, simple explanations will be sufficient.



#### NewFile

Makes a new behavior control program. You can work with multiple files of program simultaneously and the transfer/copy between two different files is possible.

#### Open

Opens existing programs for editing.

#### Save

Saves a program in progress. Do not forget to name your program when saving it for the first time.

#### Recent File

In the behavior control programmer, paths of recently opened files are saved. Therefore, you can open programs that you have recently edited without going through 'Open' and searching the files.

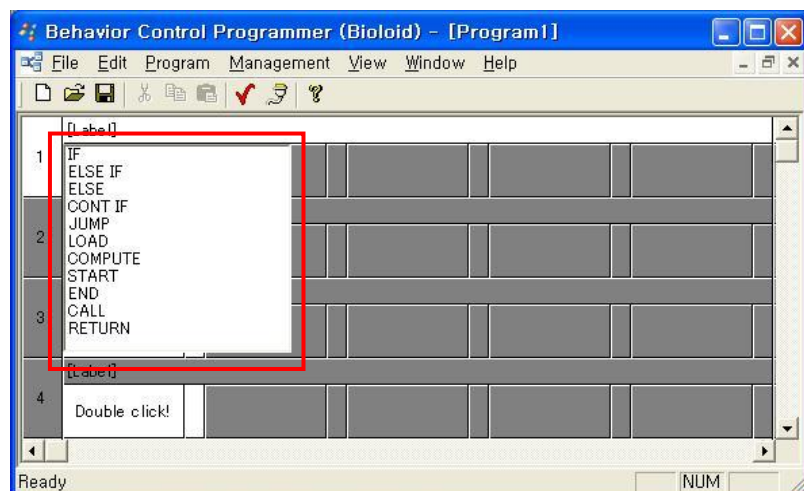


### 3-3. Grammar of Behavior Control Program

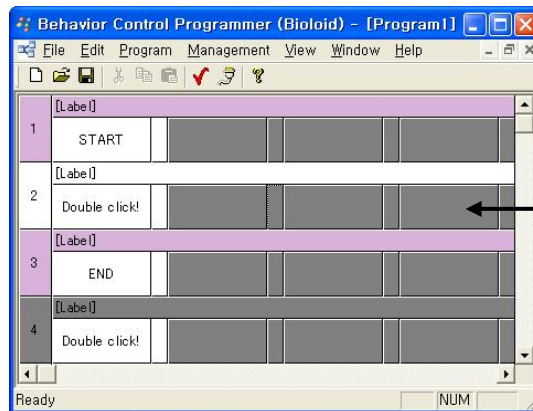
The behavior Control Program has a total of 11 commands which are classified into 3 types as follows:

Type	Command	Function
Operation	START	To indicate the beginning of a behavior control program.
	END	To indicate the end of a behavior control program.
	LOAD	To input various kinds of data.
	COMPUTE	To operate the four arithmetical operations and the logical operations.
Condition	IF	To operate the following command when the given condition is 'true'.
	ELSE IF	To come after IF and to operate the following command when the condition of IF is 'False' and the condition of ELSE IF is 'true'.
	ELSE	To come after IF or ELSE IF and to operate the command after ELSE when the both conditions of IF and ELSE IF are all 'False'.
	CONT IF	To distinguish the second IF from the first IF in the case that an IF command should be used again in another IF command.
Branch	JUMP	To move to a designated command.
	CALL	To call a designated command.
	RETURN	To return to the next row after CALL command after finishing the operation of command designated by CALL.

When you double-click the first block of a command row, you can see the above basic commands.

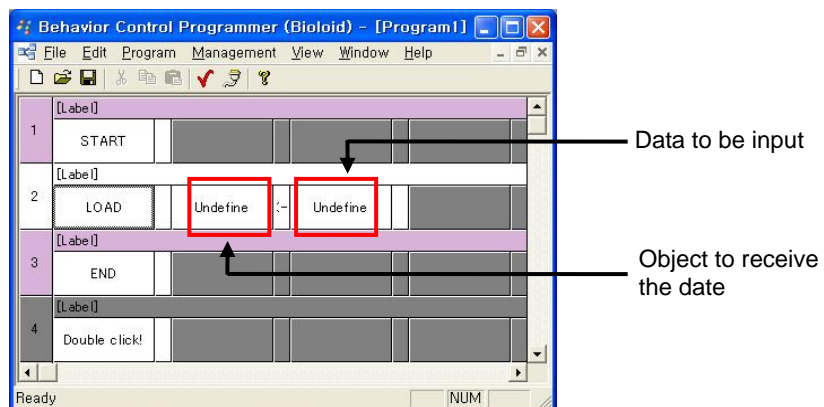


**START, END** All the behavior control programs should be made between the 'START' and the 'END' Commands.

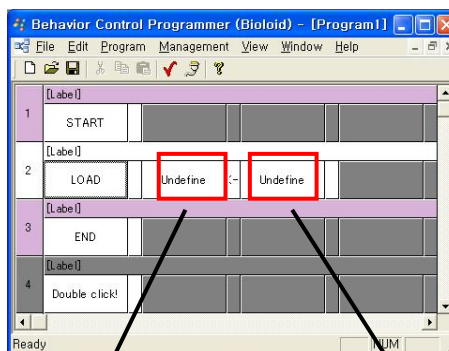


## LOAD

This is a command that inputs various kinds of data. When you select 'LOAD', two blocks next to 'LOAD' are activated as seen in the following figure. The right block is for data input while the left is for receiving data.

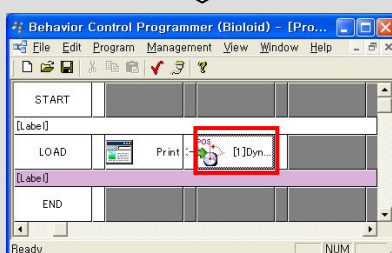
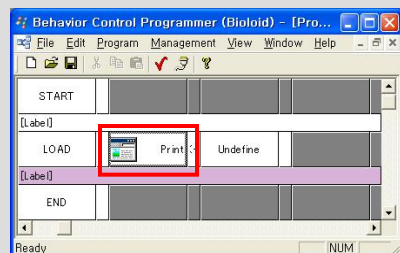
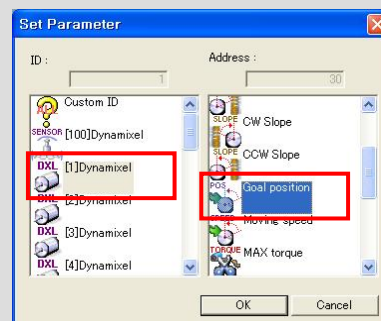
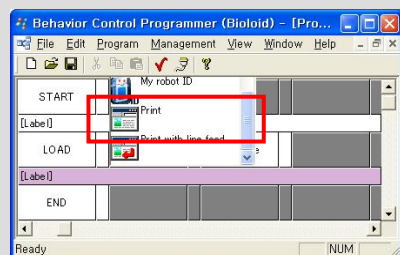
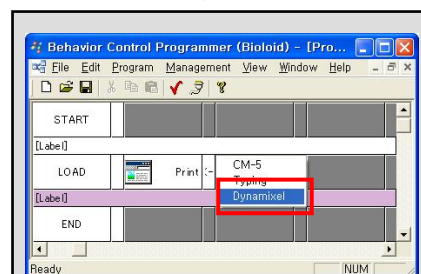
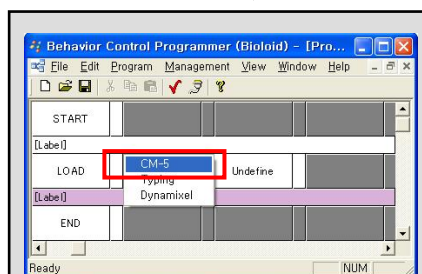


For example, if you want to see the speed of Dynamixel of number 1 on the screen, select 'Speed' in the right block and 'Print' in the left block as seen in the following figure.



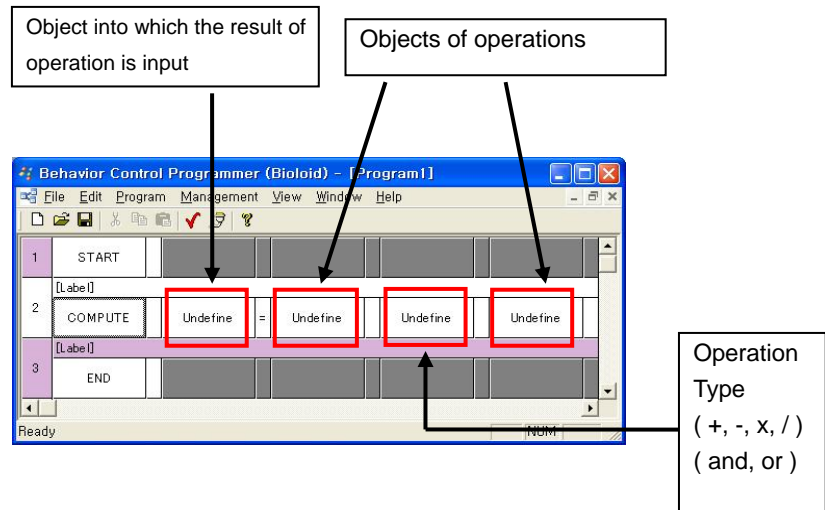
Double-Click

Double-Click



**COMPUTE**

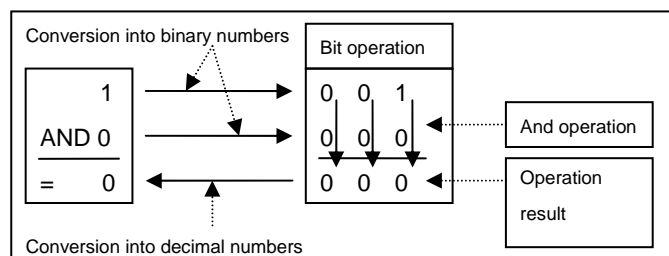
This is used to perform the four arithmetical operations ( + , - , x , / ) and the logical operations [and, or]. When you select 'COMPUTE', four blocks next to the 'COMPUTE' command are activated. The command structure is as follows.



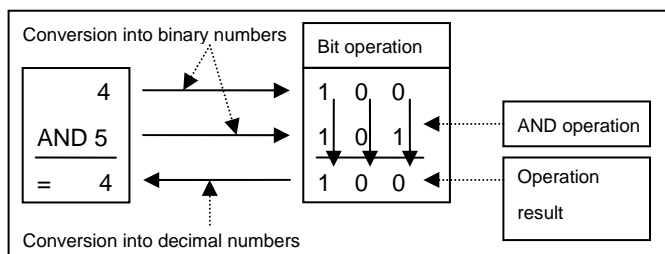
'and' and 'or' do a bit operation.

That is, you have to convert all the number you use into binary numbers. Refer to the following example.

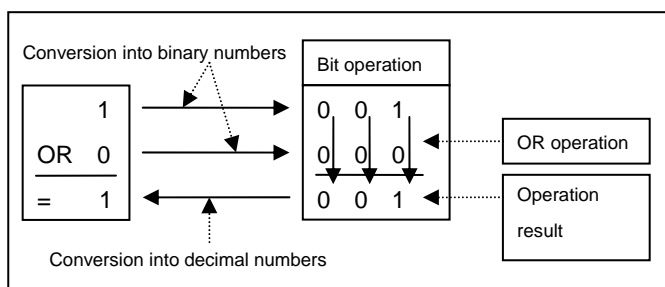
1 (001 in binary number) and 0 (000 in binary number) = 0 (000 in binary number)



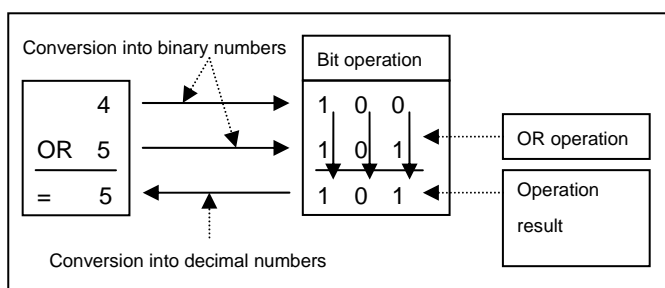
4 (100 in binary number ) and 5 (101 in binary number) = 4 (100 in binary number)



1 (001 in binary number ) or 0 (000 in binary number) = 1 (001 in binary number)

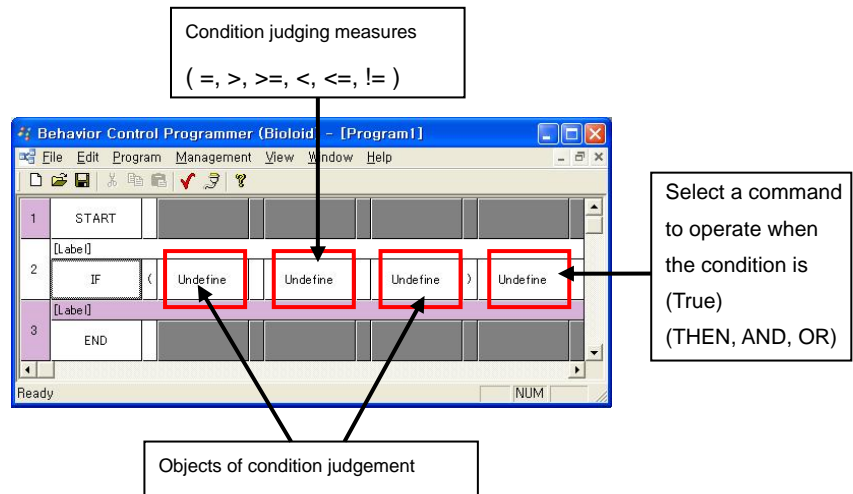


4 (100 in binary number) or 5 (101 in binary number) = 5 (101 in binary number)



**IF**

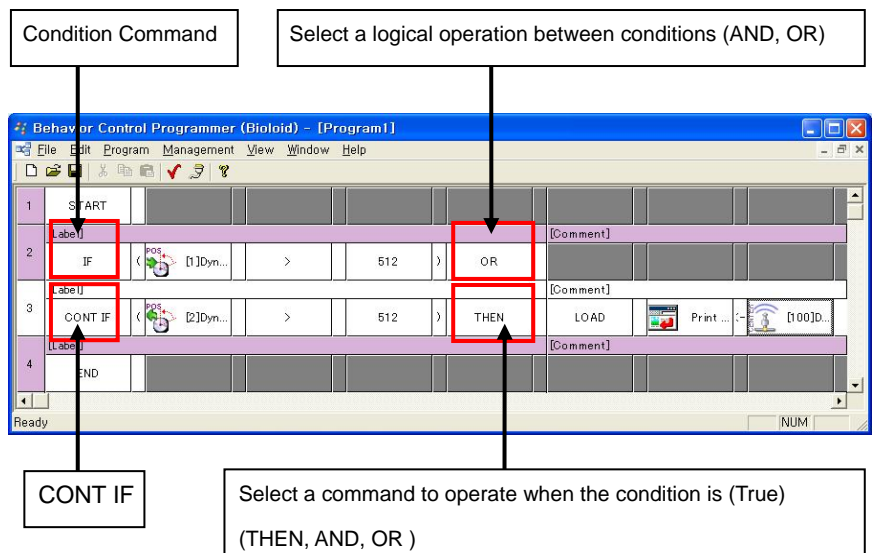
This is to judge conditions. When you select 'IF' command, 4 blocks next to this command are activated. The composition of the command is as follows.

**ELSE IF**

When there is 'ELSE IF' in the next row of IF command and the condition of IF command is FALSE, the condition command of 'ELSE IF' is executed. The command structure is the same as that of the 'IF' command.

**CONT IF**

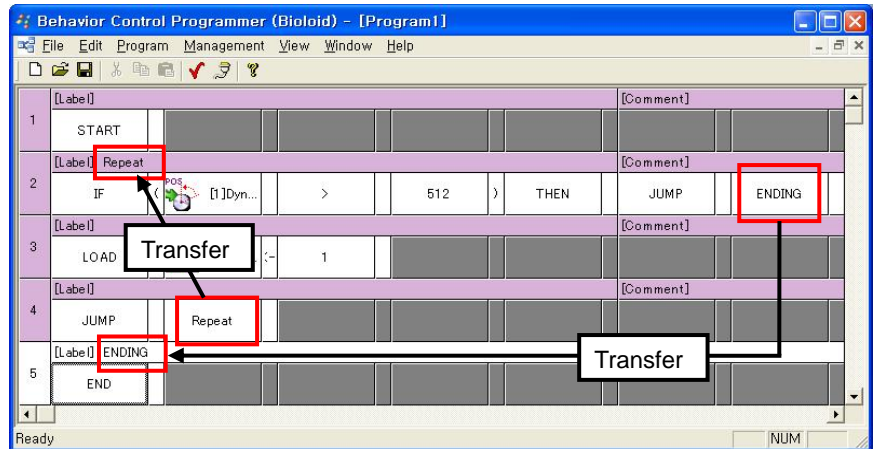
You can use this command to judge multiple conditions at the same time. The following figure displays how it works.

**ELSE**

When the conditions of the preceding commands are all [FALSE] and there is 'ELSE' in the next row, the 'ELSE' command is executed.

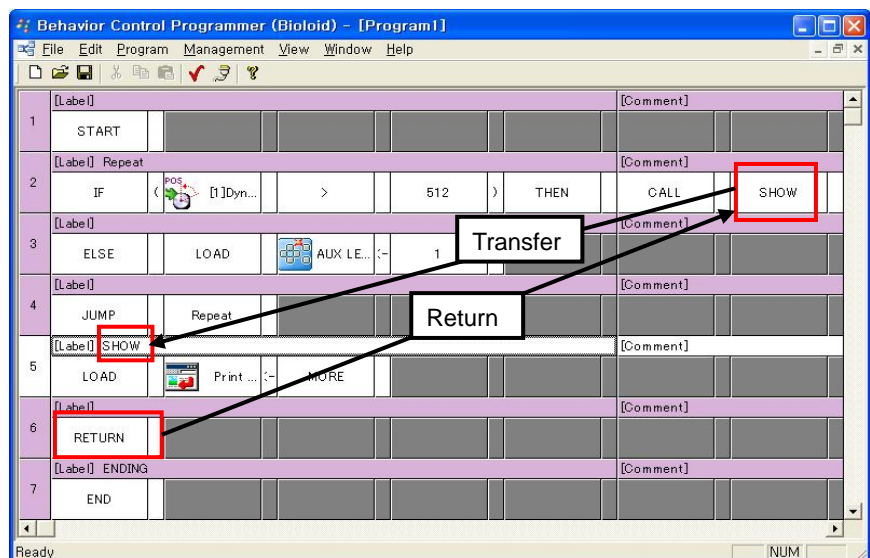
JUMP

This command changes the execution order of the commands. Input an indication of a command that you want to execute in a block next to 'JUMP'. When there is no corresponding indication, an error will occur while downloading the program. Refer to the following figure on how to use this command.



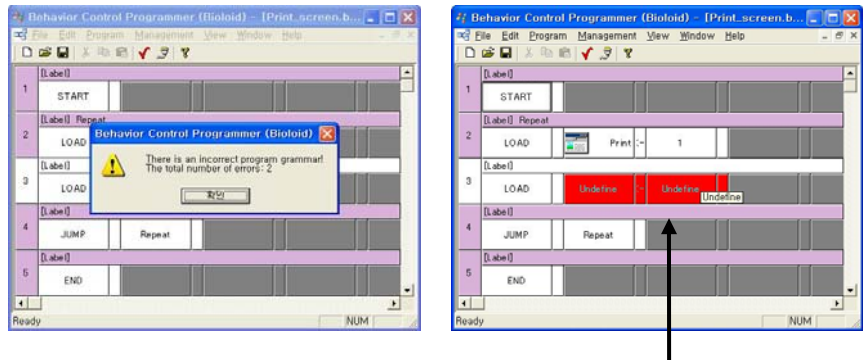
**CALL, RETURN** When you make a program, you may be required to repeat a set of commands. In this case, change a set of commands into a Sub-Routine which can be called whenever necessary. This will make the program simple and convenient to use. This 'CALL' of Sub-Routine is frequently used and is useful when the program is long.

'CALL' is similar to 'JUMP'. It differs in a way that the operation always turns back to the position where 'CALL' is executed by 'RETURN' after the sub-routine is accomplished. Sub-routine called by 'CALL' should always finish with 'RETURN'.

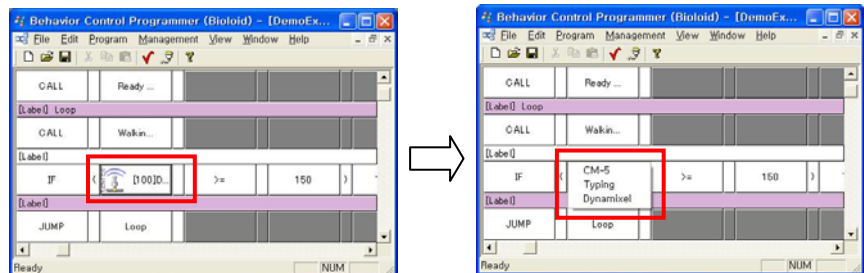


**Rule Check**

When a behavior control program is completed, you have to verify if there is no grammatical error. A behavior control program against the rule can not be downloaded into a robot. You can check errors of the program by 'RULE CHECK' in the program menu. Parts against the rule turn red. Modify these red sections to conform to the rule and repeat the RULE CHECK. [RULE CHECK] automatically runs during program downloading.

**Modification**

To add a new set of commands after the editing is finished, left-click the block to insert the new commands and then right-click to display a menu.



Click the left button of the mouse

Click the right button of the mouse

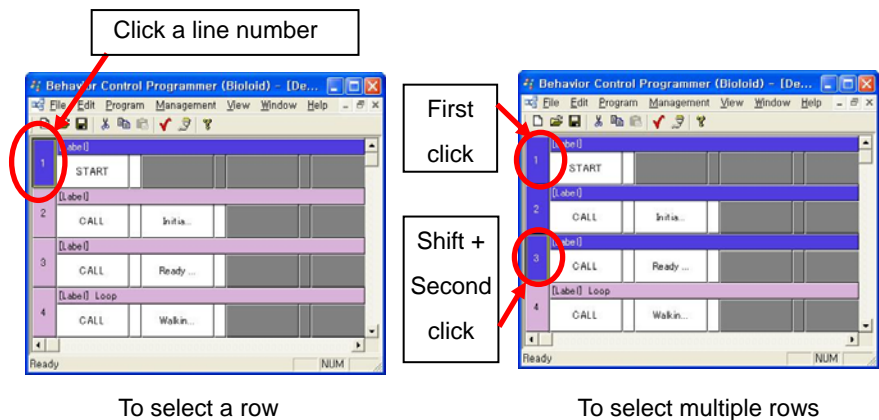


### 3-4. Edit

In the Edit menu, there are accompanying functions necessary when you make commands of a behavior control program such as Cut, Copy, Paste, Insert, Delete, Enable/Disable.

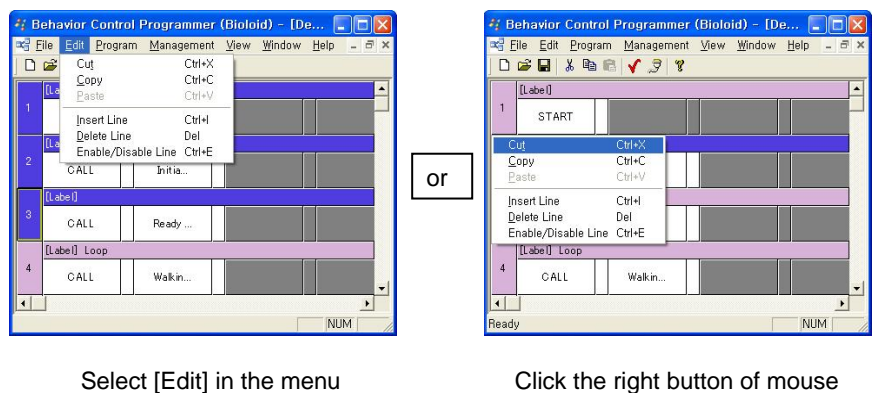
#### Selecting Command

You have to select a row of command for editing a command. When you click a line number placed in the head of the row with the left button of your mouse, the whole row is selected. If you want multiple rows at the same time, select line numbers while pressing and holding the Shift key on the keyboard.



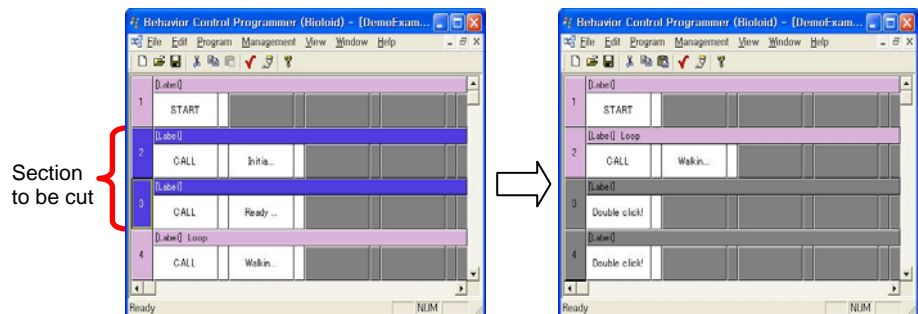
#### Editing Menu

After selecting a row of command, select [Edit] in the menu or right-click on the area of the selected command row, then the editing menu will appear.



**Cut**

[Cut] temporarily saves the selected command while deleting it from its current location. Click on [Paste] to paste the cut selection into another location.

**Copy**

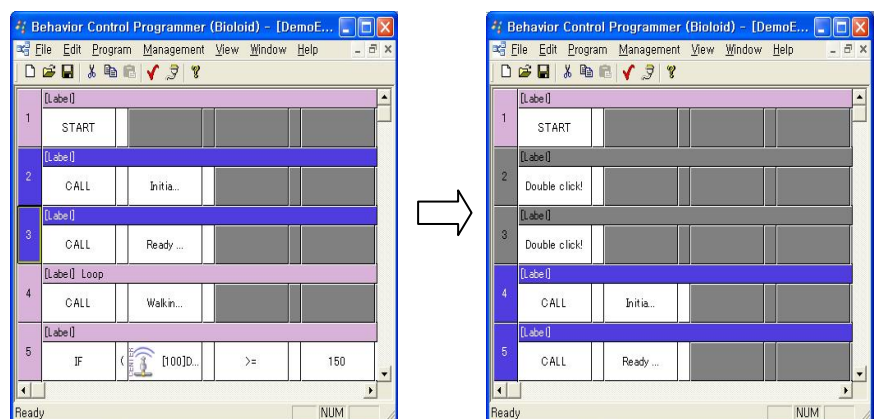
Use [Copy] to copy selected commands to a different location without deleting the original command.

**Paste**

[Paste] auto-writes commands temporarily saved by [Cut] or [Copy] into a preferred location. Commands can be transferred among behavior control programs using this function.

**Insert**

You can make a blank row using this [Insert] function. Select multiple rows as many as you want to insert and select [Insert]. You can write commands or make them by using [Paste] into the blank rows that you have inserted.

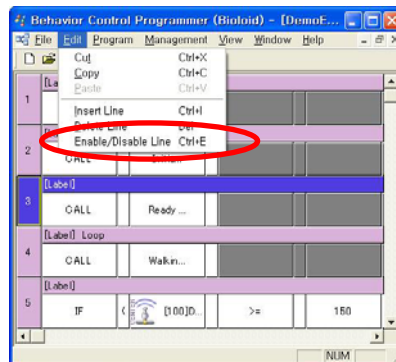


Select multiple rows as many as you want to add

Select Insert

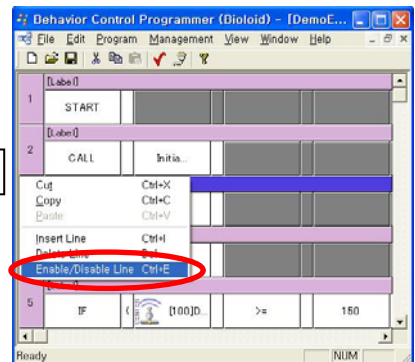
**Enable / Disable Code**

There are some occasions when you do not want to run the commands that you made but can not delete them. In this case, you can use [Enable/Disable] function. When you click [Enable/Disable], the line number and the indication tag of the selected command row turn into green and the same command is not executed. If you click again [Enable/Disable] when the [Disable] function is on, the [Disable] turns into Enable.

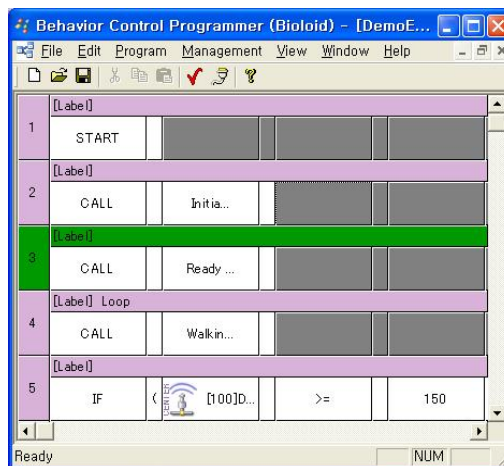


Select in the editing menu

or



Click the right button of mouse



Section to operate

Section to be disable

Section to operate

**Comment**

[Comment] are lines that contain a user's notes and does not affect the program. When you double-click a row on which [Comment] is indicated, you can have a space where you write sentences. Whatever you write in this space does not reflect on the program.



### 3-5. Running

To run a behavior control program, the program should be downloaded into the CM-2+ of the robot. Turn on the PLAY mode of the CM-2+ after download to start running the behavior control program.

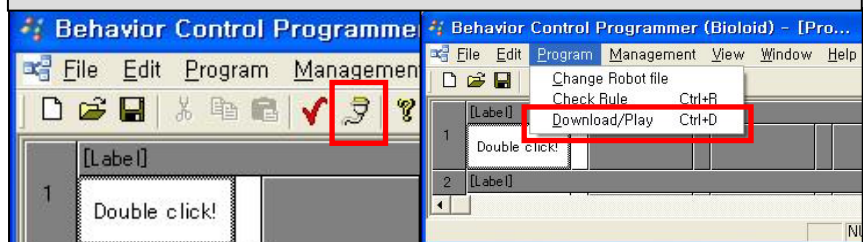
You can download the program according by:

① Connect PC and CM-2+ by a serial cable.

② Turn on the power of CM-2+.

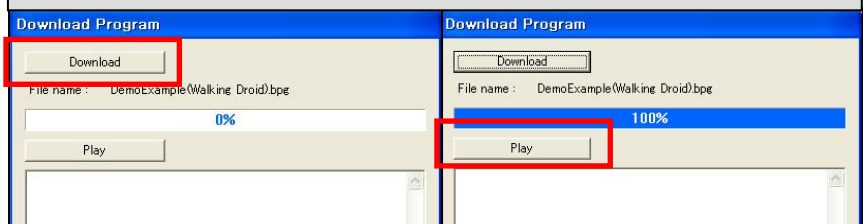


③ Select an icon for behavior control program download on the toolbar or 'Download/Play' of the program menu.



④ Click the 'Download' button in an activated window.

⑤ When the download is completed, click the 'Execute' button.



**Verification Of Connection** When the CM-2+ is not connected to the behavior control programmer, a message will appear as below.



In this case, verify the connection by checking the following in order:

- Is the serial cable well connected?
- Is the power of CM-2+ turned on?
- Is there another program using the port in operation?
- Is the port of PC correctly designated?













**PLAY** If you want to make the robot move, run the PLAY mode of CM-2+ after downloading the behavior control program.

**TIP**

We have pressed the play button and execute the program by entering the play mode while the PC was connected to the behavior control programmer. When a PC is not connected, you can execute a program by manually entering the play mode by pressing the mode change button and then pressing the start button.

### 3-6. Using CM-2+

CM-2+ input output items and their addresses

Icon	Name	Function
	Motion play page	When you input a number of a motion page, the corresponding motion of a motion program starts.
	Motion play status	1 is outputted while a motion is playing and 0 if not.
	TX romocon data	Wireless data to be sent
	RX romocon data	Received wireless data
	RX romocon data arrived	It becomes 1 when new wireless data is received and 0 when the received data is read.
	AUX LED	It turns on when 1 is inputted and turns off when 0 is inputted.
	CM-5 Button	Five buttons of CM-5 or composition of the five buttons are used as input devices of the program.
	Timer	Inputted value is reduced by 1 every 0.125 second.
	Other robot ID	Wireless ID of another robot with which you intend to communicate.
	My robot ID	Wireless ID of my robot (changeable)
	Print	Inputted value is shown on the screen.
	Print with line feed	Line is changed after inputted value is shown on the screen.

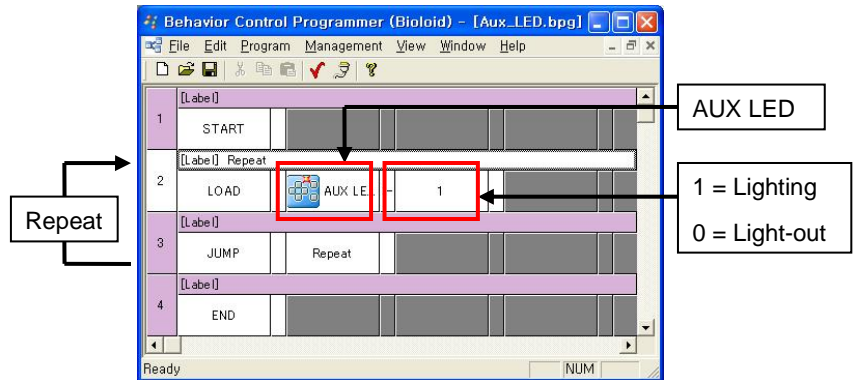
You do not have to understand all the items or memorize all the address numbers to create a program since most items have icons and names. You will learn about the items of the CM-2+ unit one by one as you read this manual. You don't have to understand all of the items for the AX-S1 and Dynamixel motors either when you first start learning about Bioloid. Just refer to the manual when necessary.

### Lighting of AUX LED

#### Example 1

Lights up the AUX distributed to users out of LED of CM-2+.

[ Step 1 ] Make a behavior control program.



[ Step 2 ] Download the behavior control program.

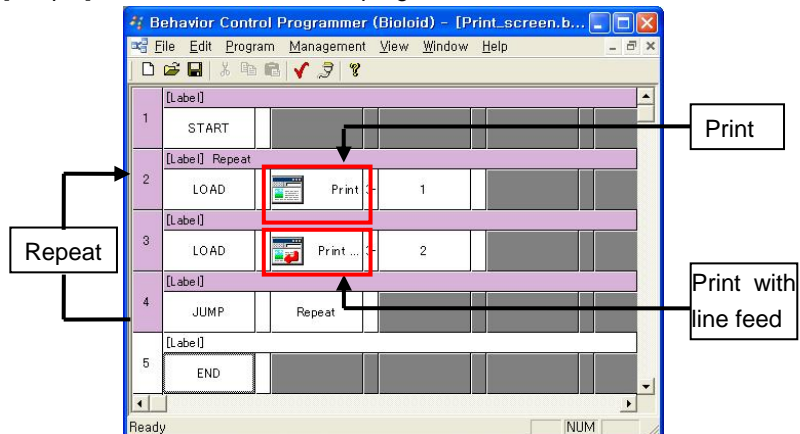
[ Step 3 ] Verify if the AUX LED of the CM-5 lights up when the 'PLAY' button is clicked on the program download window and lights out when the 'STOP' button is clicked.

### Printing on Screen

#### Example 2

Prints 1 and 2 on the screen.

[ Step 1 ] Make a behavior control program.



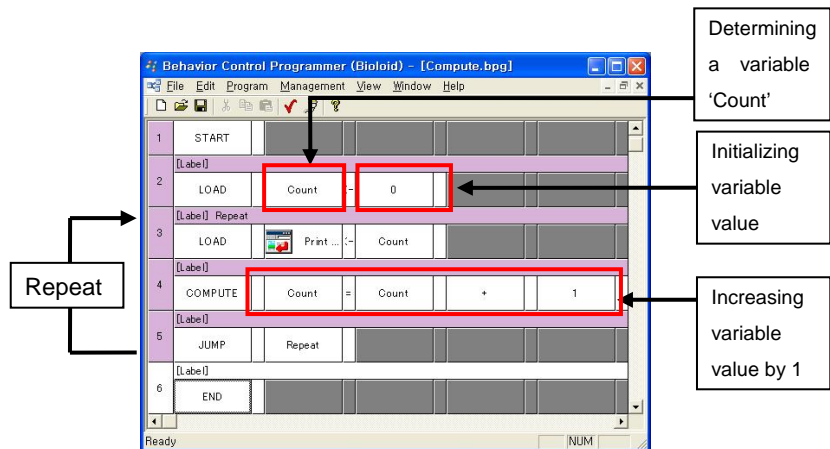
[ Step 2 ] Download the behavior control program.

[ Step 3 ] Verify if 1 and 2 are continuously printed when the 'PLAY' button is clicked on the program download window and the print is stopped when the 'STOP' button is clicked.

## Using variables

**Example 3** Prints numbers on the screen from 0 with increasing by 1

[ Step 1 ] Make a behavior control program.



[ Step 2 ] Download the behavior control program.

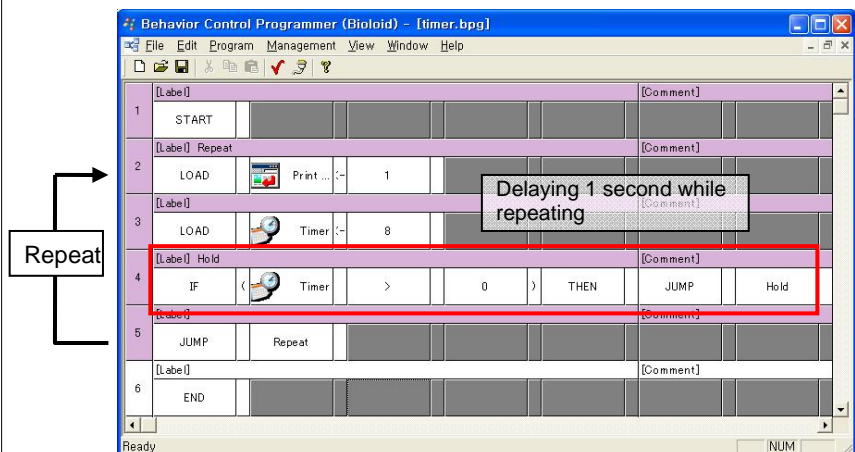
[ Step 3 ] Verify if numbers are printed on the screen starting from 1 and increasing by 1 when the 'PLAY' button is clicked on the program download window.

## Using Timer

**Example 4** Prints 1 on the screen every 1 second.

[ Step 1 ] Make a behavior control program.

Input 8 in the timer. (Timer value 8 = 1 second)



[ Step 2 ] Download the behavior control program.

[ Step 3 ] Verify if 1 is continuously printed on the screen every 1 second when the 'PLAY' button is clicked on the program download window and the print is stopped when the 'STOP' button is clicked.

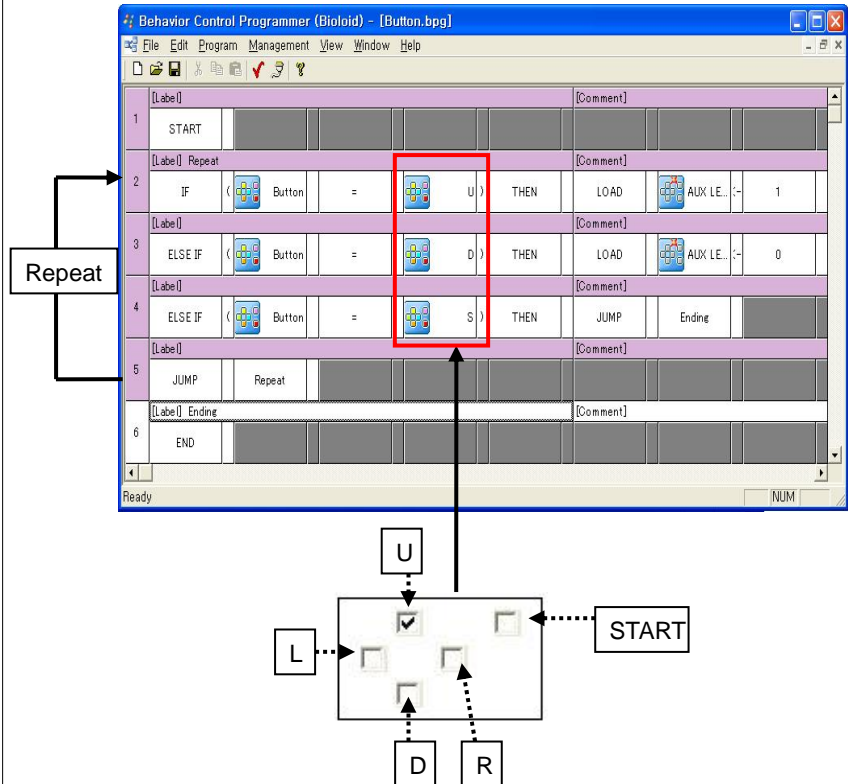


## Using Buttons

## Example 5

Makes AUX LED light up when the U button is clicked, and lights out when the D button is clicked. The operation is finished when the START button is clicked.

[ Step 1 ] Make a behavior control program.


















[ Step 2 ] Download the behavior control program.

[ Step 3 ] Verify if AUX LED lights up when U button is clicked after 'PLAY' button is clicked and LED lights out when D button is clicked on the program download window. Finish the operation by clicking on the START button.

### 3-7. Using Dynamixel

Dynamixel motor input output items and their addresses

Address	Icon	Name	Function
24		Turn on/off	When it is set as 1, power is supplied to the motor.
25		LED	It turns on when 1 is loaded and turns off when 0 is loaded.
26		CW Margin	Section made by clock wise spacing from destination position, to which torque is not applied.(0~255)
27		CCW Margin	Section made by counterclockwise spacing from destination position, to which torque is not applied.(0~255)
28		CW Slope	Section where the strength of torque is reduced as Dynamixel approaches the destination position.(CW,0~255)
29		CCW Slope	Section where the strength of torque is reduced as Dynamixel approaches the destination position.(CCW,0~255)
30		Goal Position	Joint position from 0 to 300° (Value range: 0~1023)
32		Moving Speed	Speed while moving (Value range: 0~1023)
34		MAX Torque	To determine the greatest torque (Value range: 0~1023)
36		Present position	Current position value (Value range: 0~1023)
38		Present speed	Current speed (0~1023 of values)
40		Present load	Load applied to external part (Value range: 0~1023)
42		Input voltage	Value of supplied voltageX10 (12V is read as 120)
43		Internal temperature	Internal temperature of Dynamixel(°C)
46		Moving flag	It is read as 1 while executing movement command and as 0 if not.

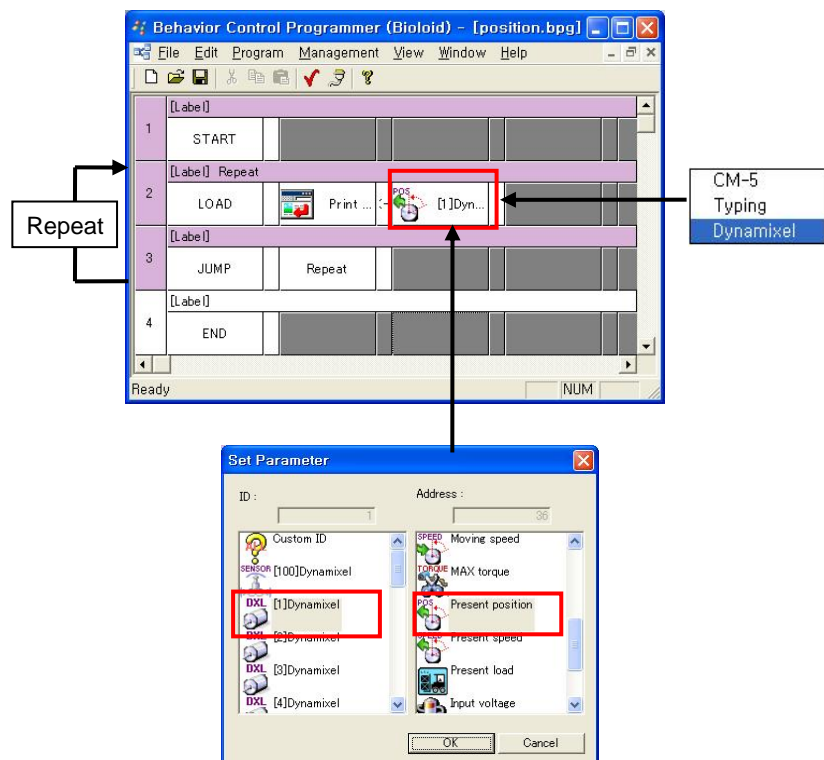
## 3-7-1. Controlling Position

**Position Value** DYNAMIXEL controls the position between 0~300° by using 0~1023 of position values. The following example helps you to understand the position values of the DYNAMIXEL.

**Example 6** Verifies the position value of each position by moving the DYNAMIXEL with hand.

[ Step 1 ] Connect the DYNAMIXEL with ID number 1 to the CM-2+.

[ Step 2 ] Make a behavior control program.



[ Step 3 ] Download the behavior control program.

[ Step 4 ] Verify changing position value with moving the DYNAMIXEL by hand after clicking 'PLAY' button on the program download window.

[ Step 5 ] Please keep the position value of each position in mind.

**Position Control** DYNAMIXEL can be moved to the desired position at a Moving Speed you want.

**Example 7** Makes a reciprocating motion between 60° and 240° at a highest speed

[ Step 1 ] Connect the DYNAMIXEL with ID number 1 to the CM-5.

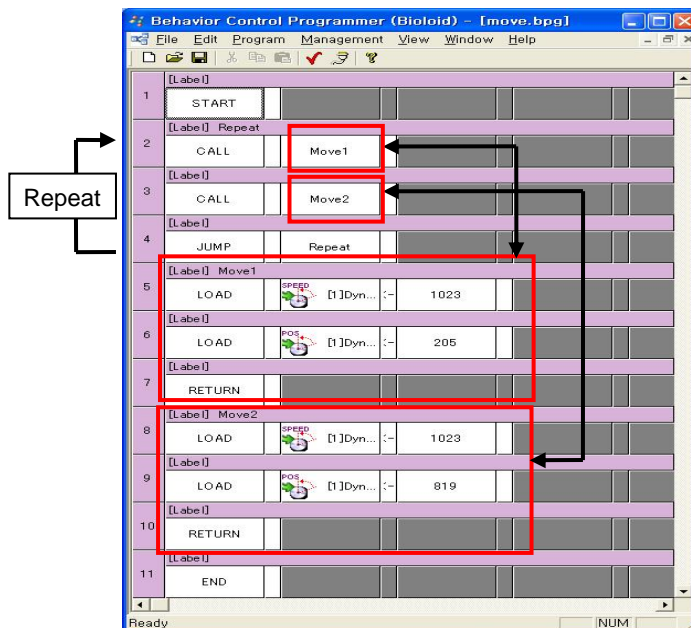
[ Step 2 ] Make a behavior control program.

① To simultaneously control the position and speed, input speed value first then position value.

(Position value: 60°= 205, 240°=819 )

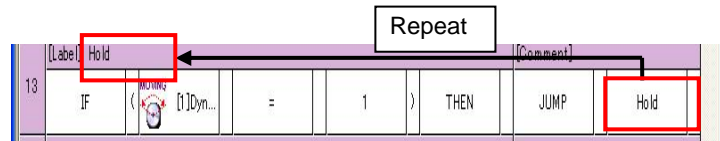


② Repeatedly call on two goal positions.

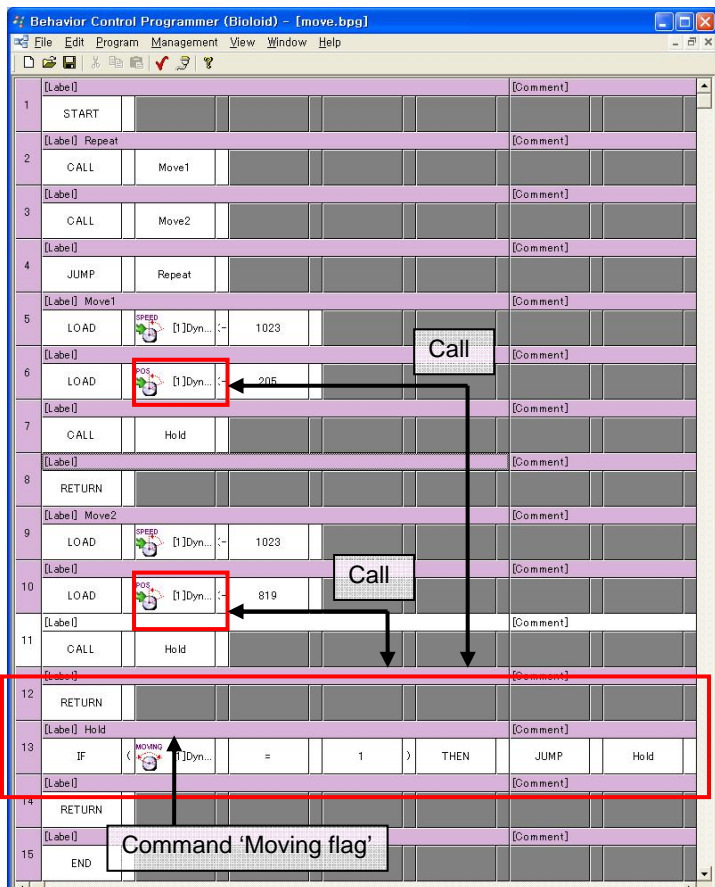


## Example 7

- ③ When you play the above program, you can see that the DYNAMIXEL does not move in the intended position. The reason is that the behavior control program which has finished the command of moving to the goal position executes another command in the next row even before the DYNAMIXEL arrives to the goal position. Therefore, a standby command to temporarily stop the program until the DYNAMIXEL arrives to the goal position is required. When you use 'Moving flag' command, you can stop the program until the DYNAMIXEL arrives to the goal position.



- ④ The completed program is as follows.



### 3-7-2. Smoothly Controlling

#### Slope

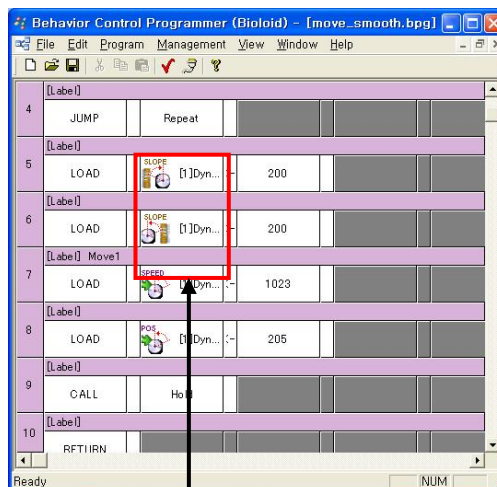
DYNAMIXEL can smoothly be moved by using the CW/CCW Slope command. Please refer to the following example on how to program.

#### Example 8

Reduces the speed as AX-12+ approaches to the goal position.

- [ Step 1 ] Connect AX-12+ with ID number 1 to CM-5.
- [ Step 2 ] Make a behavior control program.

Insert 'CW Slope' and "CCW Slope' above 'Speed' and 'Goal position' of Example 7 as following figure.



- [ Step 3 ] Download the behavior control program.
- [ Step 4 ] Verify that the speed of the AX-12+ is reduced as the AX-12+ approaches to the goal position after the 'PLAY' button is clicked on the program download window.

## Margin

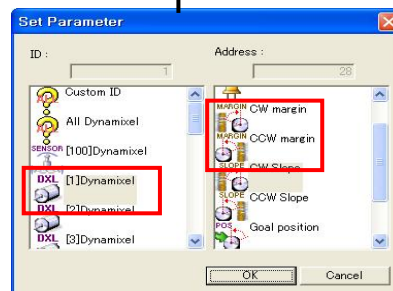
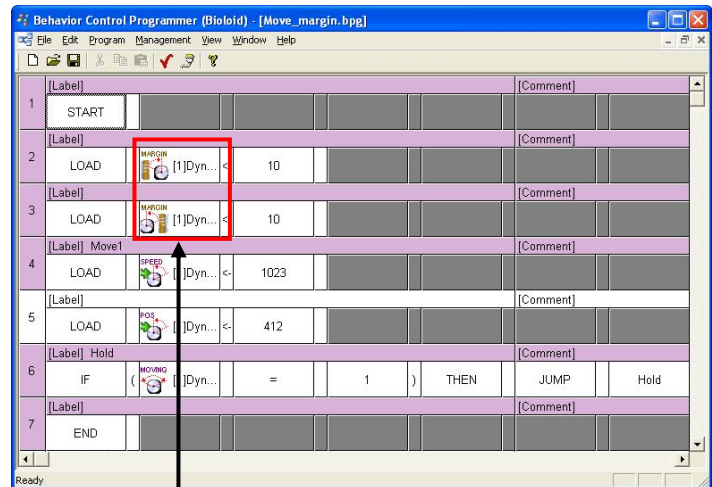
When you set the CW / CCW Margin value, a section to which torque is not applied around the Goal Position can be made. When a robot grips an object too hard, that object can be damaged. Therefore, proper force is more important than the accurate position control. If you set a proper value for Margin, the robot can grip things without damaging them.

**Example 9** Makes a section to which torque is not applied around the goal position

[ Step 1 ] Connect the DYNAMIXEL with ID number 1 to the CM-5.

[ Step 2 ] Make a behavior control program.

Insert 'CW Margin' and 'CCW Margin' above 'Speed' and 'Goal position' like following figure.



[ Step 3 ] Download the behavior control program.

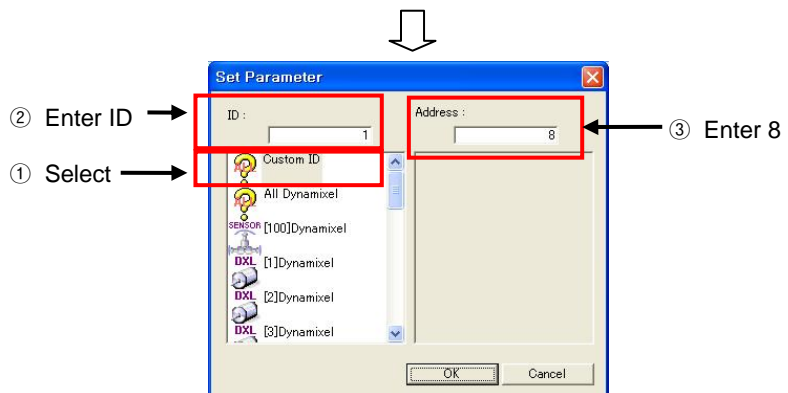
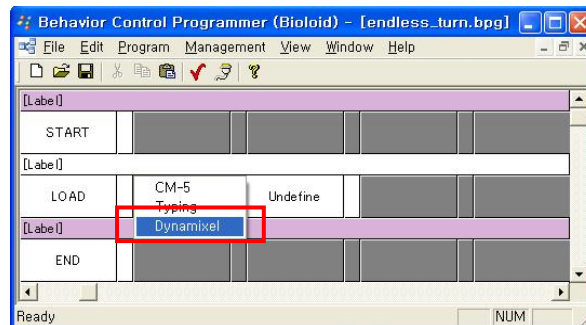
[ Step 4 ] Rotate the rotating portion by hand when the DYNAMIXEL arrives to the goal position after the 'PLAY' button is clicked on the program download window. Verify that there is a section where you can rotate the rotating portion with small force.

### 3-7-3. Endless Turn (Using the DYNAMIXEL as a wheel)

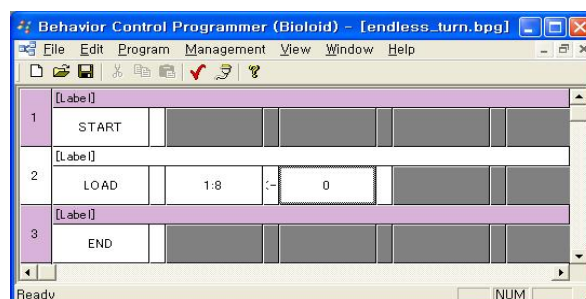
DYNAMIXEL can be used as a wheel when set to ENDLESS TURN. The example below shows how to use this setting.

#### Endless Turn Setting

When you select 'Dynamixel' of 'LOAD' in the behavior control program, an interaction window appears. Selects 'Custom ID' on the interaction window and write the ID of the DYNAMIXEL that you want to set into Endless Turn in a blank for ID. Then, input 8 in a blank for address input and click 'OK'.



When you input 0 into a command, as in the figure below, and play the behavior control program, the DYNAMIXEL with corresponding ID is set in the state of Endless Turn.







**Revolution Speed Control** You can control the speed by putting a value in the section of 'Moving Speed' in the state of Endless Turn. The larger the inputted value is, the faster the DYNAMIXEL rotates. Input 1023 for the highest speed and input 0 to stop rotation.

**Rotation / Counter-Rotation** When the input value for 'Moving Speed' is smaller than 1024, you get the desired rotation. However, when the input is larger than 1024, you get the counter-rotation. The larger the input value, the higher the counter-rotation speed. For example, if you input 600, the DYNAMIXEL rotates at a speed corresponding to 600 and if you input 1624( $600 + 1024$ ), the rotation speed will be the same but in a reverse direction.

[Rotation]

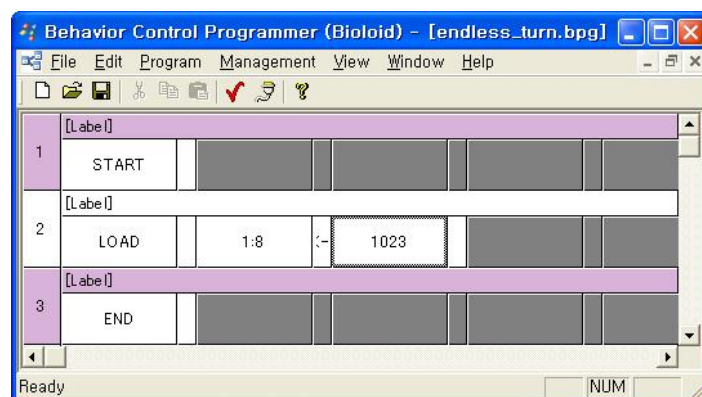
[Label]					
2	LOAD		[1]Dyn...	<-	600

[Counter-Rotation]

[Label]					
2	LOAD		[1]Dyn...	<-	1624

**Convert into** If you want to convert the state of the DYNAMIXEL into Position Control again, input

**Position Control** 1023 instead of 0 in the above behavior control program and play the program.



### Caution

Once the DYNAMIXEL is set in the state of Endless Turn, it functions in the Endless Turn until the state is changed. If you try to control DYNAMIXEL set in the Position Control state as the Endless Turn state or in reverse, DYNAMIXEL does not properly function.

## 3-8. Using Sensors

A robot cannot truly be a robot if it simply moves by remote control. A robot has to be able to move and do things autonomously. The most important thing in making a robot autonomous is to give it the ability to sense and gather information. For example, if you want to build a robot that can avoid obstacles, the robot would first need to have the ability to sense the obstacle.




















A device that can sense information is called a sensor. A sensor not only has the ability to sense objects, but also people or other robots. The process of a robot sensing outside information and reacting to it via outputting a movement is called robot interaction.

**TIP**

The interaction between a robot and human is called HRI (Human-Robot Interaction). Voice and face recognition is also part of HRI. In order to have robots live with humans, the advancement of HRI development is essential.

**Functions**

The behavior control program has the following icons and functions for controlling the AX-S1.

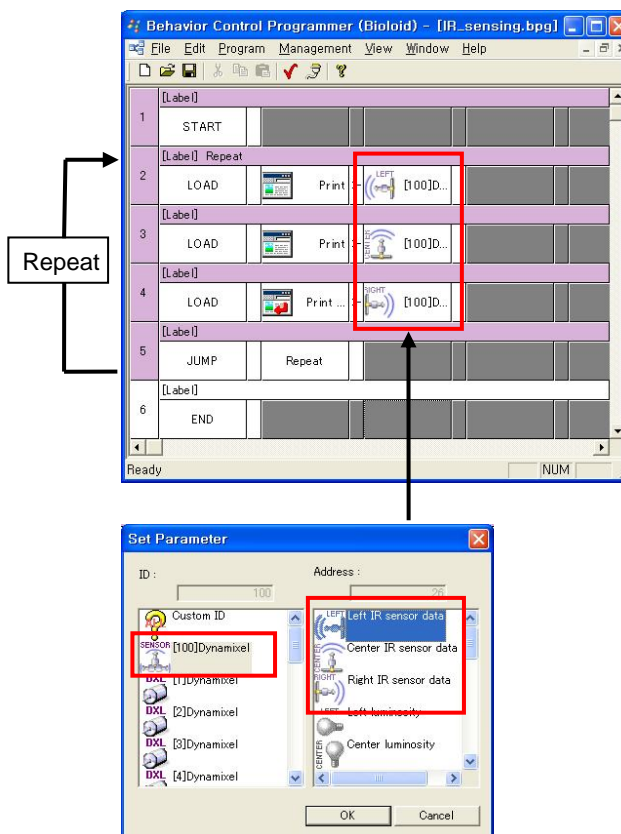
Address	Icon	Name	Function
26		Left IR sensor data	Sensing value of left side distance sensor
27		Center IR sensor data	Sensing value of center side distance sensor
28		Right IR sensor data	Sensing value of right side distance sensor
29		Left luminosity	Value of left side light brightness
30		Center luminosity	Value of center light brightness
31		Right luminosity	Value of right side light brightness
32		Obstacle sensor	To be set when the distance sensor value is larger than the standard value (bit-RCL)
33		Luminosity detection flag	To be set when the brightness value is larger than the standard value (bit-RCL)
35		Sound data	Volume of sound currently sensed
36		Sound data MAX hold	The largest value out of sound inputted up to the present
37		Sound detected count	Number of times of sound sensing such as number of times of clap
38		Sound detected time	Duration of time until sound is occurred
40		Buzzer index	0~52 (Increasing by semitone from 'La')
41		Buzzer time	0.1 second of unit, possible to (5 seconds). When the duration is 255, a sound previously inputted is played.(Buzzer scale 0~27)
46		IR remote control arrived	It is set as 1 when new data from the remote controller and it turns into 0 when the arrived data is read.
48		IR remote control RX data	Value of received remote controller data
50		IR remote control TX data	Value of remote controller data to be sent
52		Obstacle detected compare	It is a standard of data setting of address 32.
53		Light detected compare	It is a standard of data setting of address 33.

## 3-8-1. Distance Sensing

**Distance Sensing Value** AX-S1 has three infrared sensor(IR Sensor). The closer things are, the larger the distance sensing value is. The following example should help you understand the distance sensing value of the AX-S1.

**Example 10** Verifies that the distance sensing value changes when a hand approaches the AX-S1.

- [ Step 1 ] Connect the AX-S1 with ID number 100 to the CM-5.
- [ Step 2 ] Make a behavior control program.



- [ Step 3 ] Download the behavior control program.
- [ Step 4 ] Verify that the distance sensing value changes when a hand approaches the AX-S1 after 'PLAY' is clicked on the program download window.
- [ Step 5 ] Keep the value of distance sensing according to the approaching distance in mind.

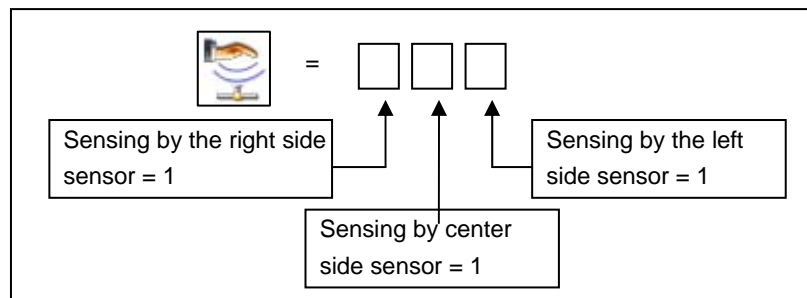
**Judgment based on**

**Obstacle Sensing** There are occasions when sensing the presence of an obstacle is more important than sensing its distance while the robot operates. In this case, you can use 'Obstacle detected compare' and 'Obstacle detection flag'.

'Obstacle detection flag' generates values as follows when an obstacle is sensed within a distance value designated by 'Obstacle detected compare'. The following values are bit ones, that is, binary numbers.

For example, when the left side IR sensor senses an obstacle, the value of 'Obstacle detection flag' is 001 and when the center side IR sensor senses an obstacle, the value of 'Obstacle detection flag' is 010. The value of 'Obstacle detection flag' is 011 when the left side IR sensor and the center side one sense obstacles at the same time.

The following table presents values in decimal number used in the behavior control program converted from values in binary number generated by 'Obstacle detection flag'.



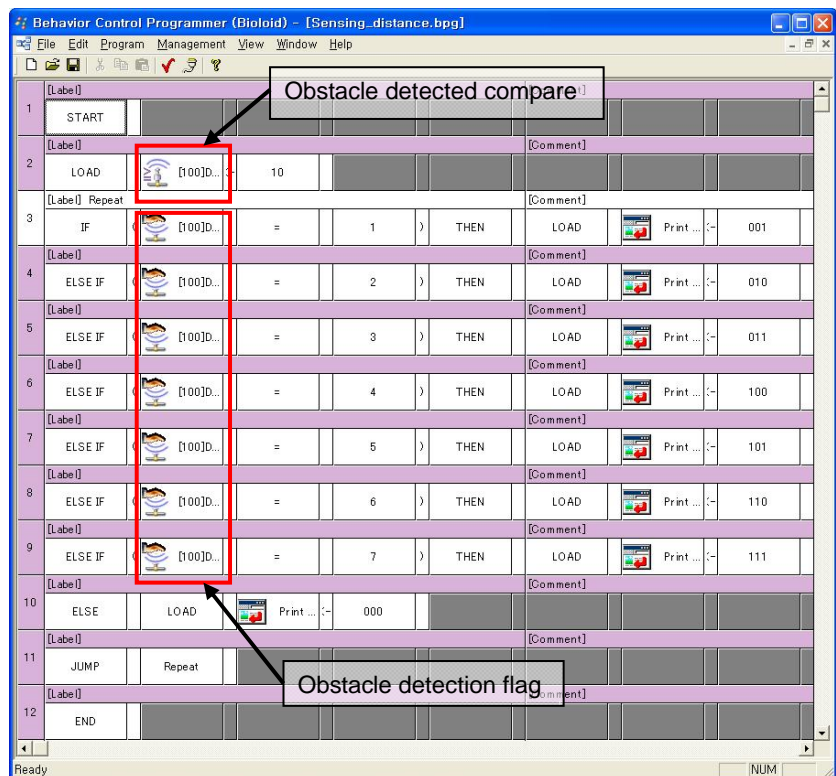
Value in binary number	Value in decimal number	Meaning that [Obstacle Sensor] interprets
000	0	No obstacle is sensed.
001	1	The left side sensor senses an obstacle.
010	2	The center side sensor senses an obstacle.
011	3	Both the left side sensor and the center side one sense obstacles.
100	4	The right side sensor senses an obstacle.
101	5	Both the right side sensor and the left side one sense obstacles.
110	6	Both the right side sensor and the center side one sense obstacles.
111	7	The sensors of all the sides sense obstacles.

The following example verifies that the 'Obstacle detection flag' displays the direction where obstacles are sensed.

**Example 11** Outputs number '1' in a place corresponding to the direction where a hand is sensed.  
( Left=001, Center=010, Right=100 )

[ Step 1 ] Connect the AX-S1 with ID number 100 to the CM-5.

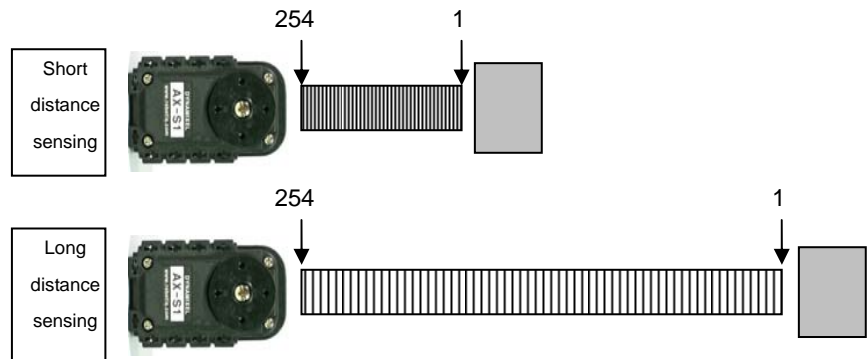
[ Step 2 ] Make up a behavior control program as follows.



[ Step 3 ] Download the behavior control program.

[ Step 4 ] Verify that the output values change with approximating a hand from various directions to the AX-S1 after the 'PLAY' button is clicked on the program download window.

**Short distance/ Long distance sensing** There are two types of distance sensing of the AX-S1; short distance sensing and long distance sensing. You can get more accurate distance values by the short distance sensing and expand farther the sensing area by the long distance sensing.



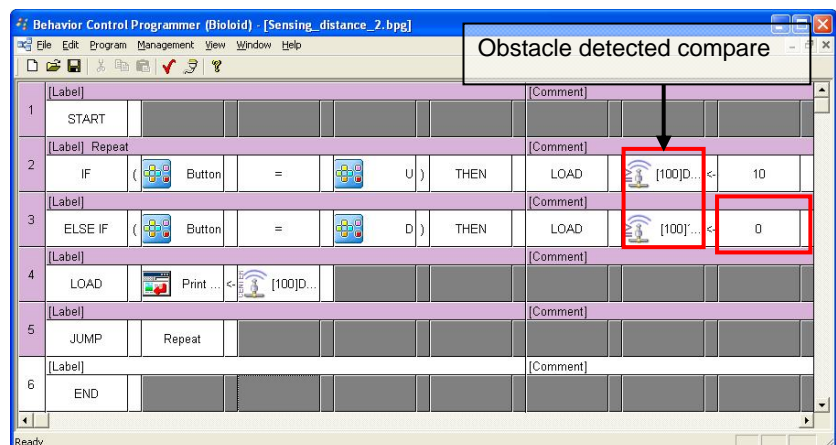
Set 'Obstacle detected compare' as 0 for using the short distance sensing. The following example verifies the difference between two types of sensing.

#### Example 12

When the **U** button of the CM-5 is clicked, the long distance sensing functions, and when the **D** button is clicked, the short distance sensing functions.

[ Step 1 ] Connect the AX-S1 with ID number 100 to the CM-5.

[ Step 2 ] Make behavior control program.



[ Step 3 ] Download the behavior control program.

[ Step 4 ] After the 'PLAY' button is clicked on the program download window, press the U button of the CM-5 then verify that values of distance sensing change with approximating a hand to the AX-S1. Press the D button of the CM-5 and then verify that values of distance sensing change with approximating a hand to the AX-S1.

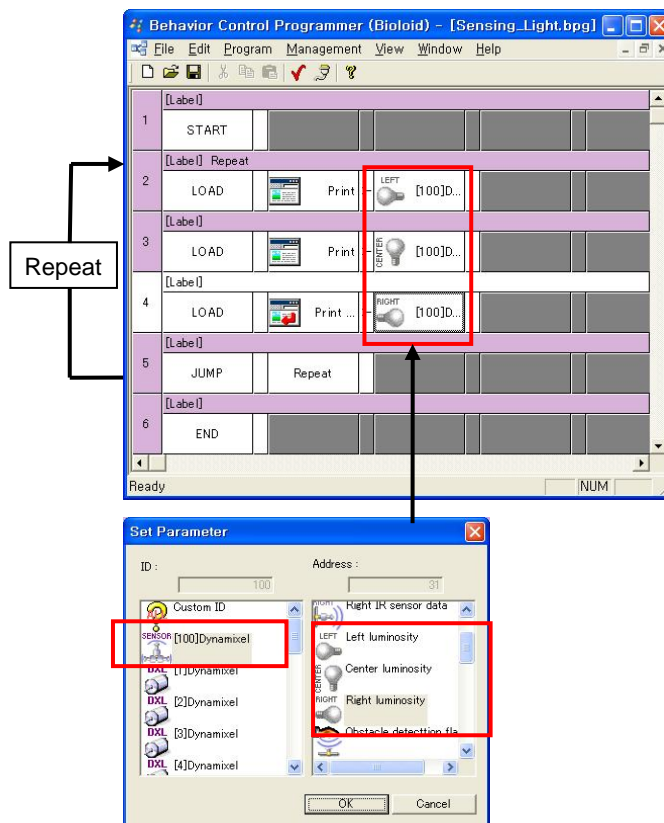
### 3-8-2. Brightness Sensing

**Value of Light Brightness** AX-S1 has three sensors for sensing brightness in its surroundings. The brighter the lights are, the larger the values of luminosity. The following example should help you understand the value of luminosity of the AX-S1.

**Example 13** Verifies that the value of luminosity changes with flashing a bright light to the AX-S1.

[ Step 1 ] Connect the AX-S1 with ID number 100 to the CM-5.

[ Step 2 ] Make a behavior control program.



[ Step 3 ] Download the behavior control program.

[ Step 4 ] After the 'PLAY' is clicked on the program download window, verify that the value of luminosity changes with flashing a bright light such as a lamp light to the AX-S1.



## 3-8-3. Sound Sensing

The AX-S1 can sense sounds in its surroundings and has several functions.

**Value of Sound data:** When there is no sound, the value is about 128. The louder the sound, the closer the value to 255. Sound data inputs about 3800 times per second.

**Sound data MAX hold:** It shows the largest sound value out of the sensed sound.

**Sound detected count:** This is a function of counting by each time when a sound louder than a certain volume such as claps is sensed. However, once a single sound is counted, the counting does not function for 80 msec to prevent the misinterpreting of one clap as several claps. The below example explains the above function.

**Example 14** Verifies that Sound data value changes by making noises such as claps toward the AX-S1.

[ Step 1 ] Connect the AX-S1 with ID number 100 to the CM-5.

[ Step 2 ] Make a behavior control program.

The screenshot shows the Behavior Control Programmer (Bioid) interface. The program is titled "Sensing\_sound.bpg". It contains 11 steps:

- START
- CALL init
- IF (Button = U) THEN CALL init
- LOAD Print [100]D...
- LOAD Print MAX [100]D...
- LOAD Print [100]D...
- JUMP Repeat
- LOAD MAX [100]D... 0
- LOAD [100]D... 0
- RETURN
- END

A red box highlights steps 4, 5, and 6. A "Repeat" label with an arrow points to step 7, indicating a loop from step 7 back to step 3.

[ Step 3 ] Download the behavior control program.

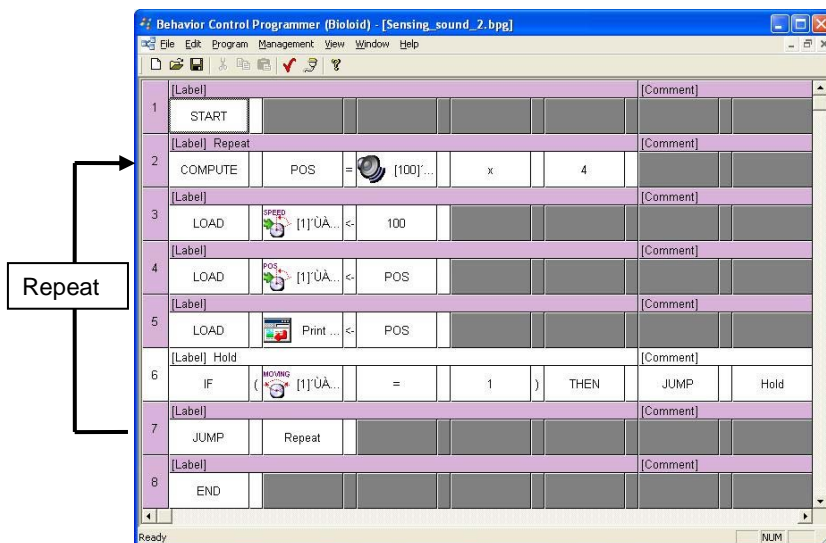
[ Step 4 ] After the 'PLAY' button is clicked on the program download window, verify that the values of Sound data, Sound data MAX hold, and Sound detected count change by making noises such as claps toward the AX-S1. Verify also that the values are initialized when the U button is pressed.

**Example 15** Making the AX-12+ move more as the surrounding sound becomes louder.

[ Step 1 ] Connect the AX-S1 with ID number 100 and AX-12+ with ID number 1 to the CM-5.

[ Step 2 ] Make a behavior control program.

- ① Make a variable called 'position value' and input a value of 'Sound data' into this variable. Since the range of 'Sound data' value is 0~255 and the range of 'goal position' value is 0~1023, quadruple the 'Sound data' value so that this value becomes similar to the 'goal position' value.
- ② Input the variable 'position value' as a 'goal position' value of the AX-12+ with ID number 1. Determine an appropriate value for a speed value.



[ Step 3 ] Download the behavior control program.

[ Step 4 ] After the 'PLAY' button is clicked on the program download window, verify that the AX-12+ moves more as the sound becomes louder by making noises such as claps toward AX-S1.

### 3-8-4. Melody Playing

AX-S1 has a Buzzer that enables it to make sounds. You can make interesting robots that sing or express their emotion by using this melody playing function.

**Value of Buzzer index**      AX-S1 can make 52 notes.

Buzzer value	note	Buzzer value	note	Buzzer value	note	Buzzer value	note
0	La	15	Do	27	Do	39	Do
1	Ra#	16	Do#	28	Do#	40	Do#
2	Si	17	Re	29	Re	41	Re
3	Do	18	Re#	30	Re#	42	Re#
4	Do#	19	Mi	31	Mi	43	Mi
5	Re	20	Fa	32	Fa	44	Fa
6	Re#	21	Fa#	33	Fa#	45	Fa#
7	Mi	22	Sol	34	Sol	46	Sol
8	Fa	23	Sol#	35	Sol#	47	Sol#
9	Fa#	24	La	36	La	48	La
10	Sol	25	La#	37	La#	49	La#
11	Sol#	26	Si	38	Si	50	Si
12	La					51	Do
13	La#						
14	Si						

**Duration of Buzzer Sound**      AX-S1 has a function of regulating the Buzzer time. The value 1 of the buzzer sound duration is 0.1 second. That is, The value 10 of the buzzer sound duration means 1 second. The minimum duration is 0.3 seconds and the maximum duration is 5 seconds.

When you input 254 in 'Buzzer time' and a buzzer note value that you want in 'Buzzer index', the buzzer keeps sound without stopping. To stop the buzzer sound, input 0 in 'Buzzer time'.

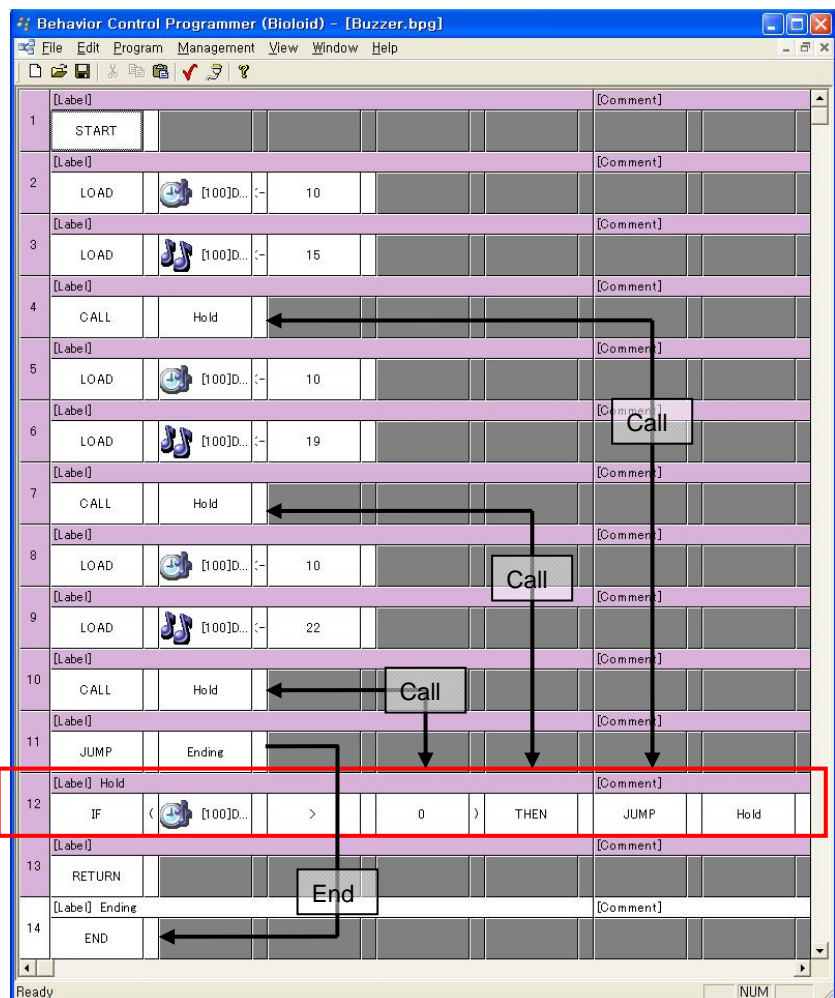
The following demonstrates how to sound the buzzer.

**Example 16** Playing notes of Do, Mi, and Sol. Each note is played for one second.

[ Step 1 ] Connect the AX-S1 with ID number 100 to the CM-5.

[ Step 2 ] Make a behavior control program.

To play a note for as long as you want, the behavior control program has to stop operating for a while until the play finishes. 'Buzzer time' becomes 0 when the play finishes. Make a program as below using 'Buzzer time' as 'Moving flag' of Example 7.



[ Step 3 ] Download the behavior control program.

[ Step 4 ] After 'PLAY' button is clicked, verify that the notes of Do, Mi, and Sol is played each for 1 second.

**Playing Particular Melodies** The AX-S1 has 27 melodies composed of various notes. Set 'Buzzer time' to 255 to play one of those melodies.



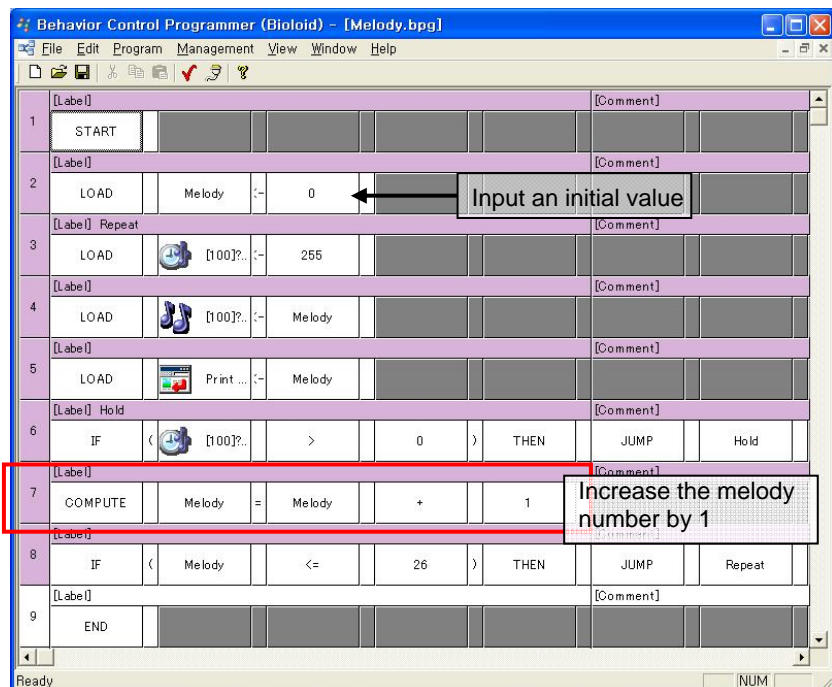
When you input one of values between 0 and 26 in 'Buzzer index', the corresponding melody is played.



#### Example 17 Plays saved melodies one after the other.

[ Step 1 ] Connect the AX-S1 with ID number 100 to the CM-5.

[ Step 2 ] Make a behavior control program.

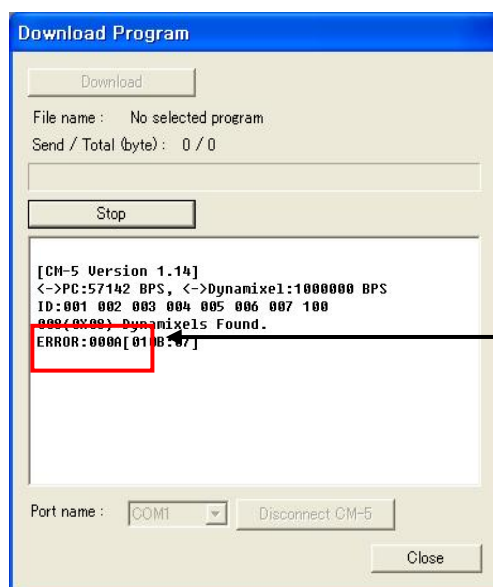


[ Step 3 ] Download the behavior control program.

[ Step 4 ] Verify that melodies are played one after the other after clicking the 'PLAY' button on the program download window.

### 3-9. Error Code

If the behavior program has an error, the error message comes out on the download program window like below. In this case, please edit your program referring to the table below.



Error Code

Code	Meaning	Check Point
8100	Access to the unavailable Dynamixel ID	Check whether an using ID of AX-12+ or AX-S1 that does not exist in robot
		Check the cutting line
0009	Using too many call commends	Check whether using more than 9 Call commends in series
000A	Using the Call / Return incorrectly	Check whether using Return Commend without Call Commend
		Check whether using the combination of Jump Commend and Return Commend, instead of the combination of Call Commend and Return Commend
8001	Access Error ( Reading )	Check whether using an unavailable reading address of CM-2+
8002	Access Error ( Writing )	Check whether using an unavailable writing address of CM-2+



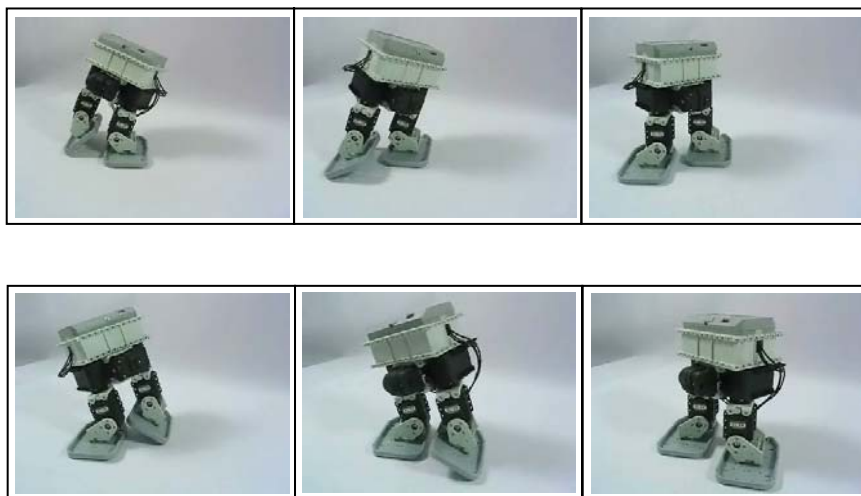
## 4. Motion Programming

Motion is a set of actions that the user assigns to a robot. Setting up the motions of a robot is called Motion Programming.

It is possible to embody simple motions using the behavior control program but it is more efficient to use the motion programming when you want a robot do complex actions. You can save complex motions in the CM-2+ which can be called by the behavior control program.

There are two ways of motion programming. One is using the motion editor and the other is running the program mode in the robot terminal. The motion editor is composed of graphics so that beginners can easily use it. As for the robot terminal, since it is composed of texts, you can see all the information as numerical values, convenient only for advanced users.

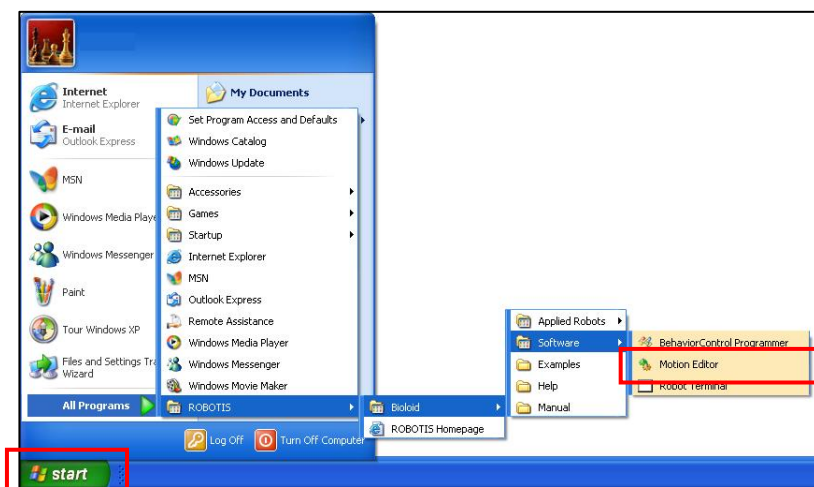
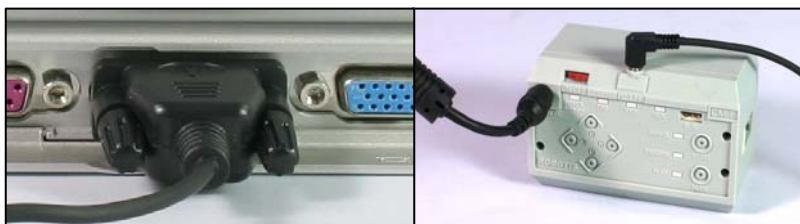
This chapter shows the methods of motion programming by using the motion editor composed of graphics.



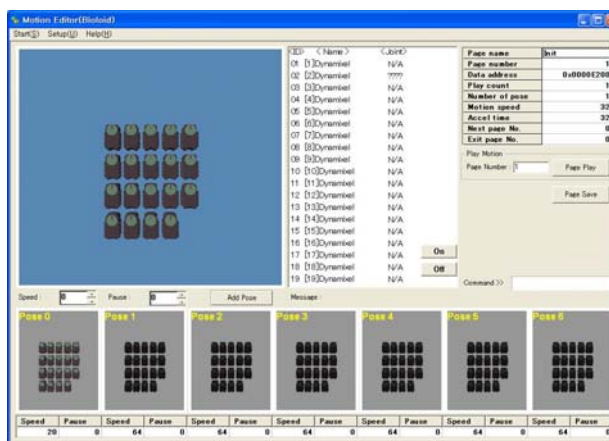


## 4-1. Starting Motion Editor

**Connection** As shown in the following figure, connect the CM-2+ to your PC. Turn the power on and select 'Start >> All programs >> Robotis >> Bioloid >> Software >> Motion Editor'. This will start the motion editor.



Initial Page of the Motion Editor



### In case of connection failure

**Connection failure** To make the motion editor function, the connection of PC and the CM-2+ of the robot should be made while the CM-2+ is in standby mode. If the connection fails, select 'Start >> CM-5 connection' in the motion editor menu and properly set the communication port[COM port]. If the connection still fails,, check if the corresponding communication port is used by another program.

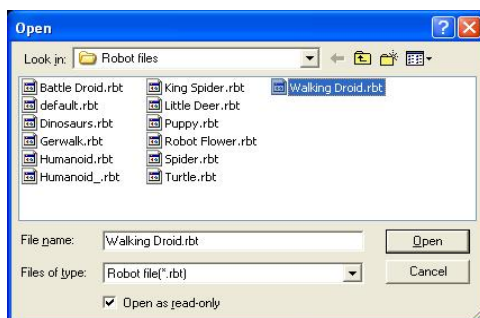
## Robot Profile

When the motion editor is functioning properly, select the type of robot that you made in the ‘Setting’ menu as in the following figure.

- ① Select 'Changing Robot Information' in the 'Setting' menu.

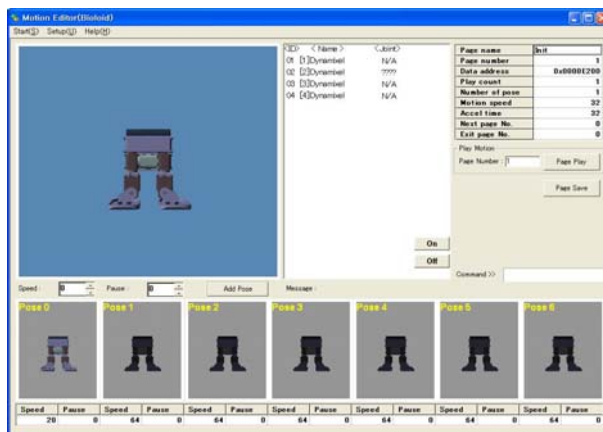


- ② Select the type of robot that you made in the appeared interaction window.



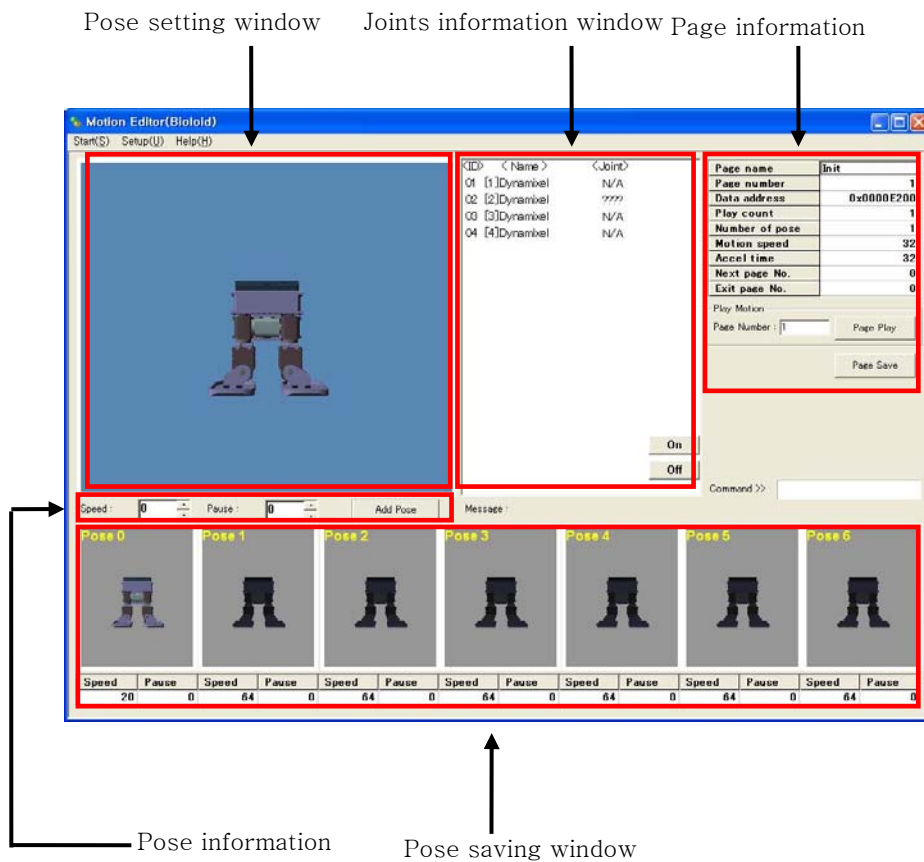
- ③ Since the robot information is applied on the next run of the program, finish the motion editor before conducting a re-run.

When you select ‘Walking Droid.rbt’, the motion editor runs as follows.



### Composition of Motion Editor

The motion editor is composed of the below. Each composing item will be explained in detail in the next chapter.



## 4-2. Creating Motions

### 4-2-1. Page Information

The page information window contains items needed to execute motions. You can modify each item by double-clicking the item number.

Page name	Init
Page number	1
Data address	0x0000E200
Play count	1
Number of pose	1
Motion speed	32
Accel time	32
Next page No.	0
Exit page No.	0

Play Motion

Page Number :

**Page Name** Determines the page name. The initial value is 'No name'.

**Page Number** This is the proper serial number for each page. CM-2+ can save and make 127 motion pages. The page number uses values from 1~127.

**Page change** Modifies the page number to move to another page.

**Data Address** Shows the memory location of the current page. General users do not need to pay attention to it.

**Play Count** Sets the number of times a motion in a page is executed. The initial value is 1.

**Number of Pose** Executing a motion, only the poses from 0 to the designated number are played. Poses beyond these numbers are displayed in black on the screen. This function can be useful when playing some of the forepart among the inputted poses. [The pose will be explained in detail in the next chapter.]

**Motion Speed** Regulates the speed at which a page is played. The initial value is 32. Changing this value into 64 doubles the motion speed.

**Acceleration Time** When a motion is being executed, all the joints work on the condition that acceleration → maintenance of speed → deceleration are repeated. Here, the running time of acceleration/deceleration is 'acceleration time'. If the acceleration time is greatly reduced, a sudden acceleration will be made to reach the designated speed which could damage the joints. On the contrary, if acceleration time is increased, there could be a section that may not move.

**Next Page No.** Since only 7 poses can be made in one page, use another page when you want to make more than 7 poses. Input the page numbers to be played successively. When you do not designate the next page, it is set as 0. [The initial value is 0.] If you input the current page number in this section, the motion is infinitely repeated. To stop this motion, click the 'Stop' button.

**Exit Page No.** Designates a page to be run when the motion play is stopped. When the 'Stop' command for stopping the motion is input by the behavior control program as well, the exit page is played first and then the motion is stopped. [To give the 'Stop' command through the behavior control program, input '0' in 'LOAD >> CM-2+ >> Robot Motion' command.] When this function is not necessary, put 0 in this section. [The initial value is 0.]

**Play Motion** When you click 'Page Play' button, the poses shown in the pose saving window are performed in sequence. When the motion is played, 'Page Play' button is changed into 'Stop' button. When you click the 'Stop' button, the motion currently performed is stopped. If you input another page number in the 'page number' section and click the 'Play' button, the motion of another page is played.

**Save Page** Since the motion data in editing is kept in the RAM of the CM-2+, it is deleted when you change pages. You have to click the 'Save Page' button to save the motion data.

## 4-2-2. Making Poses

## Pose

A pose is an intermediate movement for making a motion. For example, you need to make several poses to make a motion. A walking droid makes several poses to execute a single step. This can be compared to the films of a movie. You can make up to 7 poses in 1 page. When you need to use more than 7 poses, use additional pages by using 'Next Page No.'



**Making up poses** Please refer to the below for steps on creating poses. The image in the pose setting window is the current pose of the robot.

① Click 'Off' in the joints information window to remove force

ID	< Name >	< Joint >
01	[1]Dynamixel	????
02	[2]Dynamixel	????
03	[3]Dynamixel	????
04	[4]Dynamixel	????

② Move the joints of robot by hand to make an intended movement.

③ Click 'On' in the joints information window to fix the joints.

ID	< Name >	< Joint >
01	[1]Dynamixel	0401
02	[2]Dynamixel	0396
03	[3]Dynamixel	0441
04	[4]Dynamixel	0327

Pose setting window

**Minute Adjustment Of Pose** When you click on the <joint value>, you can minutely adjust the joint value. If you select numbers of <joint value> while you press the Ctrl Key and click ON or OFF button, only the selected joints will be ON or OFF.

Minute Adjustment of joint value

<ID>	< Name >	<Joint>
01	[1]Dynamixel	0401
02	[2]Dynamixel	0396
03	[3]Dynamixel	0441
04	[4]Dynamixel	0327

Click

On/Off of particular joint

<ID>	< Name >	<Joint>
01	[1]Dynamixel	0401
02	[2]Dynamixel	<input checked="" type="checkbox"/>
03	[3]Dynamixel	0441
04	[4]Dynamixel	<input checked="" type="checkbox"/>

Ctrl+Click =&gt; On / Off

**Pose Speed**

You can set the movement speed of pose by setting the pose speed value.

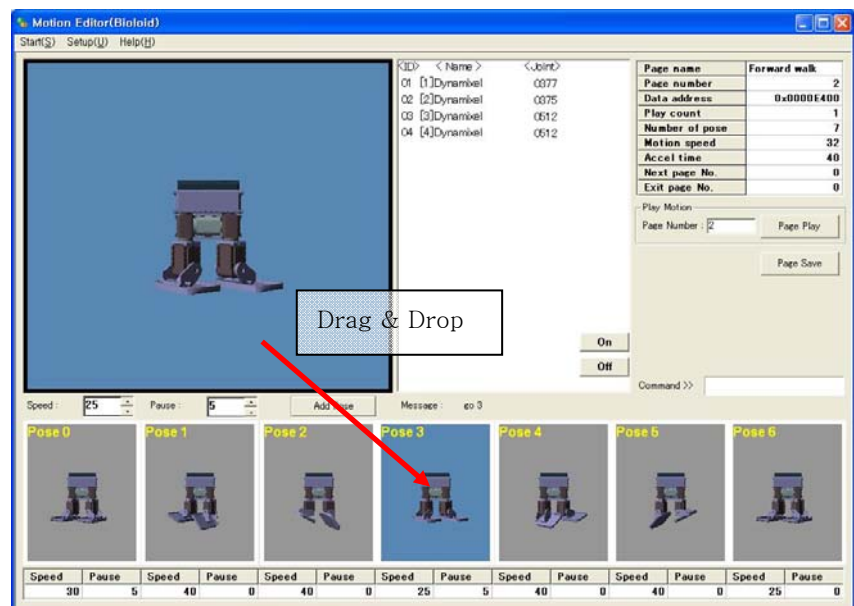
**Pause Time**

You can make the motion pause in a pose that you want. It pauses for a period of Pause Time value X 7.8 msec.

Speed :	<input type="text" value="10"/>	Pause :	<input type="text" value="50"/>
---------	---------------------------------	---------	---------------------------------

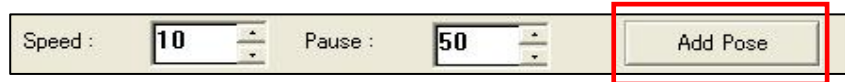
**Save Pose**

A motion is saved poses performed in sequence. Therefore, poses made, using the above process should be saved in the pose saving window. You can input the current pose in a preferred place of the pose saving window by using 'Drag & Drop' with the mouse. If you drag and drop a new pose into a place where another pose already exists, the existing pose will be replaced by the new pose and will be deleted.

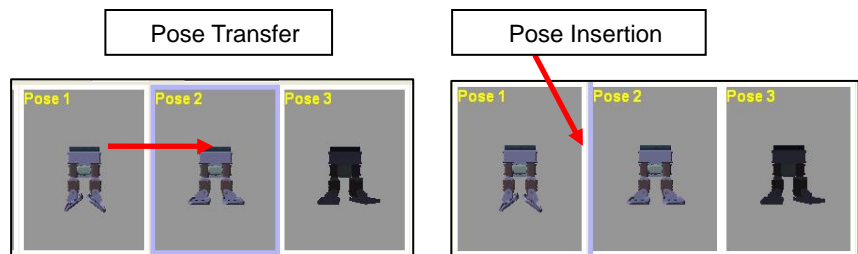


**Add Pose**

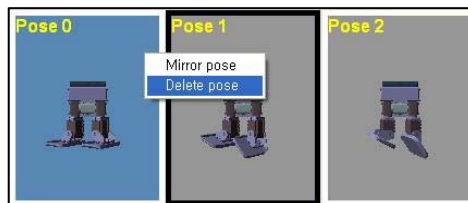
When you click 'Add Pose', the current pose is saved in the last.



**Pose Transfer, Insertion** Click and drag a pose that you want to transfer into a preferred location. If you put this dragged pose between poses, the pose is inserted in-between. This function can also be used when you transfer a pose in the pose setting window.

**Erase**

Put the mouse on a pose you want to delete, click the right button of the mouse, then, select 'Erase Pose'.

**Pose Play**

Play a pose saved in the pose saving window by double clicking on a pose. To prevent the robot from being damaged due to running inappropriate poses, the program will ask you to verify when the pose is not executable. You can determine available poses by the color of the robot in the pose window. When the pose is not executable, the robot color turns black.

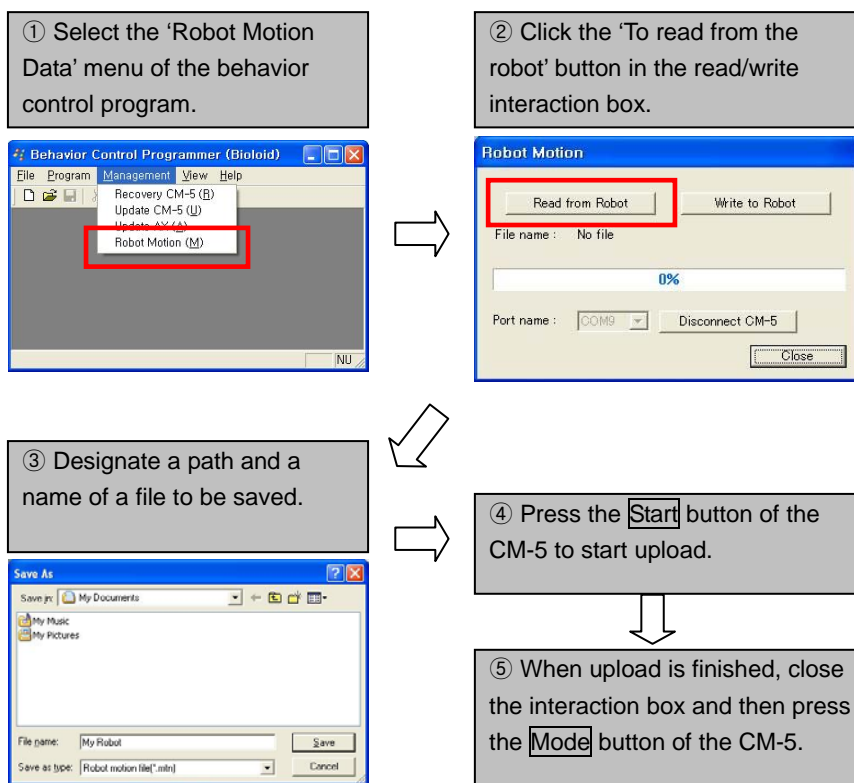




### 4-2-3. Uploading/Downloading Motion Data

The motion data made by the motion editor is saved in CM-2+ of the robot. The motion data of the robot should be uploaded and saved in the PC. In addition, the motion data in the PC should be downloaded in the robot. This function can be useful when you want to keep the motion data of the robot as a backup file in the PC or when you want several robots to use one motion data together. Upload/Download is performed in the behavior control program.

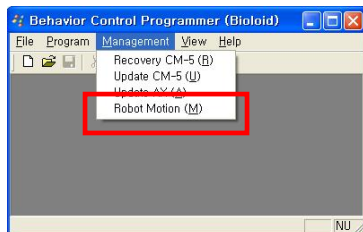
**Uploading Motion Data** Run the behavior control program and save the motion data of the robot in the PC by using the following method. The extension of the motion data file is .mtn.



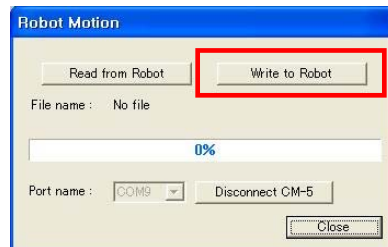
**Downloading  
Motion Data**

Run the behavior control program and download the motion data from the PC to the robot using the following method.

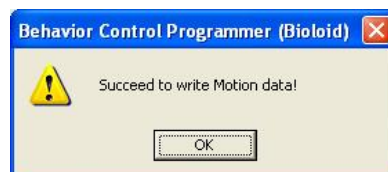
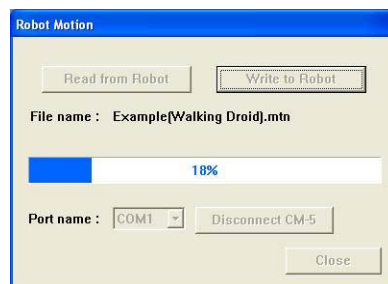
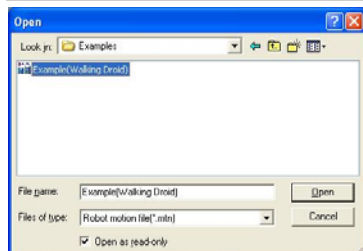
① Select the 'Robot Motion Data' menu of the behavior control program.



② Click the 'To write to the robot' button in the read/write interaction box.




③ Select a file from the PC to be saved in the robot.



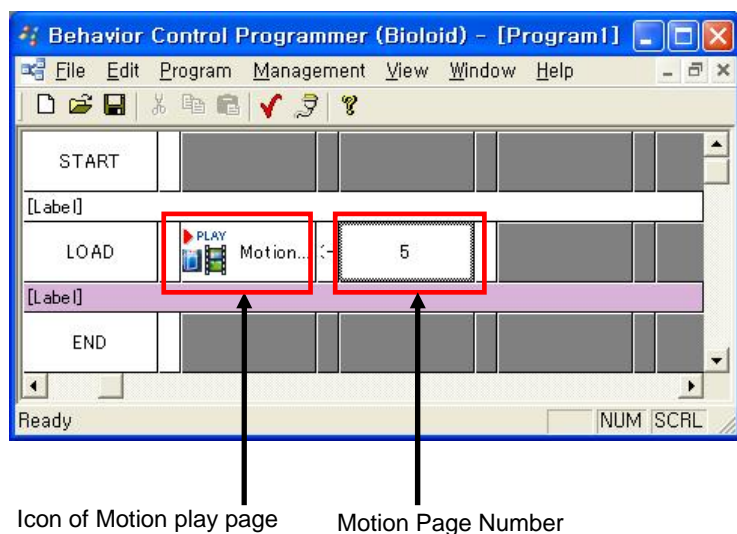
### 4-3. Playing Motion

The motion created in the motion editor is saved in the CM-2+ of the robot. To use this motion, the motion program of the CM-2+ should be executed through the behavior control program. You can play the robot motion by using the following two icons of the behavior control program.

Icon	Name
	Motion play page
	Motion play status

**Asynchronous** Use the 'LOAD' to play motions. When you select 'LOAD', two blanks are

**Motion Play** activated. Select 'CM-2+ > Motion play page' on the left. On the right one, input the 'Motion page number' of the page where the motion is located. When you make up the behavior control program as below, download it in the CM-2+ , and execute the 'PLAY' mode, the motion saved in the page numbered 5 is performed.



The following is a programming example for making the 'Walking Droid' walk.

**Example 18** Makes up a program to control the walking direction of the 'Walking Droid' by using the direction buttons of the CM-5.

[ Step 1 ] Assemble a walking droid referring to the Quick Start.

[ Step 2 ] Run the motion program and program 5 walking motions as follows.

Page Number 1 : Standby motion

Page Number 12 : Moving forward motion

Page Number 13 : Moving backward motion

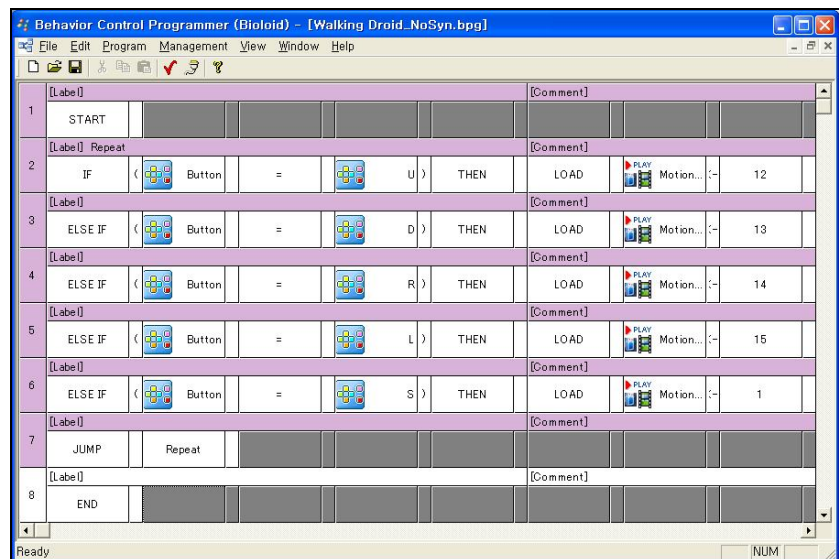
Page Number 14 : Right turn motion

Page Number 15 : Left turn motion

※ You can use a program already made for the above motions.

Download the same from 'C:\Program Files\Robotis \Boiloid\Applied Robots\Beginner\Walking Droid \ DemoExample(Walking Droid).mtn'.

[ Step 3 ] Make a behavior control program as below to perform the above motions by receiving commands inputted through buttons of the CM-5.



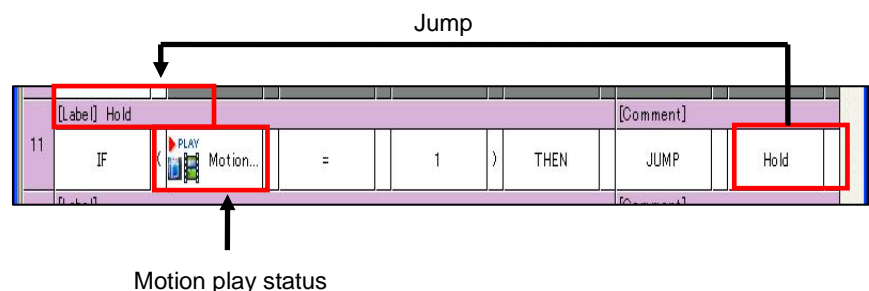
[ Step 4 ] Download the behavior control program to the robot and run the PLAY mode on the CM-5.

### Synchronous Motion Play

The motion play of the robot is executed by the behavior control program giving orders to the motion program. However, after giving an order for motion play, the behavior control program executes a command in the next row even before the motion is finished. Therefore, to make a command for continually performing more than two motions, you should make a command to make the next motion performed at the point that the first motion is finished. This is the synchronous motion play.

For this synchronous motion play, use the 'Robot Motion Status'. This command becomes 1 while the motion is performed and 0 when the motion is finished. So, you make a behavior control program to pause when the 'Robot Motion Status' is 1 and to continue to proceed when the 'Robot Motion Status' is 0.

If you make a behavior control program like below, you can make the program stop for a while.



The following is an example of a walking droid operated by the IR sensor. The robot moves backward when an obstacle is detected in the front.

**Example 19** Programming a walking motion of walking droid that moves backward when an obstacle is detected while it moves forward.

[ Step 1 ] Assemble a walking droid referring to the Quick Start.

[ Step 2 ] Run the motion program and program the following two walking motions.

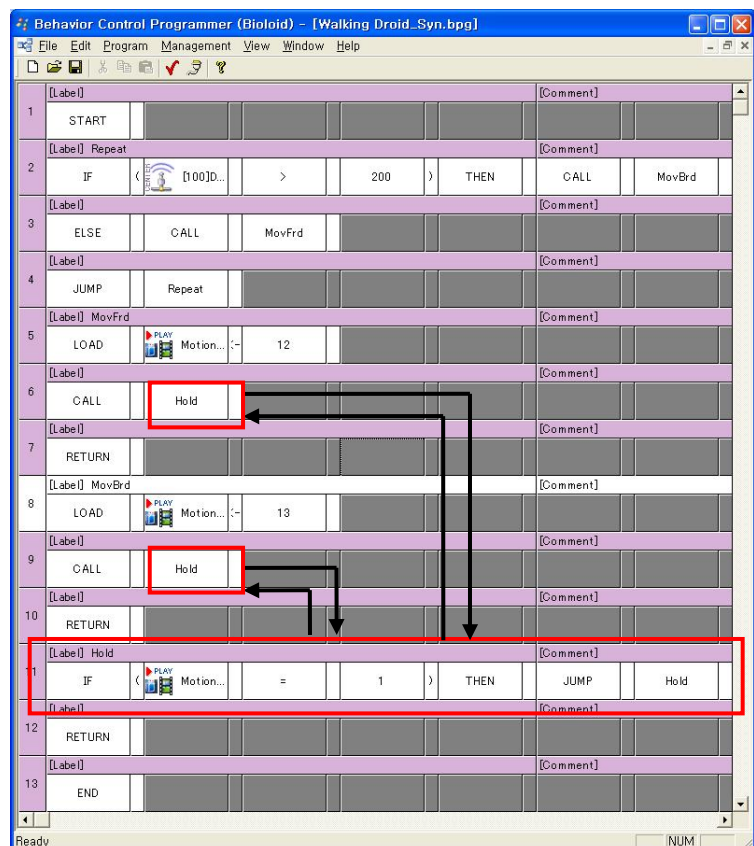
Page Number 12 : Moving forward motion

Page Number 13 : Moving backward motion

※ You can use a program already made for the above motions.

Download the same from 'C:\Program Files\Robotis \Boiloid\Applied Robots\Beginner\Walking Droid \ DemoExample(Walking Droid).mtn'.

[ Step 3 ] Make a behavior control program like below to perform the above motions by receiving commands input through the IR sensor of the AX-S1.



[ Step 4 ] Download the behavior control program to the robot and run the PLAY mode on the CM-5.

**Execution of Infinite** Motions regularly repeated like walking become efficient when they are made

**Repetition Motion** with infinite repetition motion. To make infinite repetition motion, just input your own page into the next page.

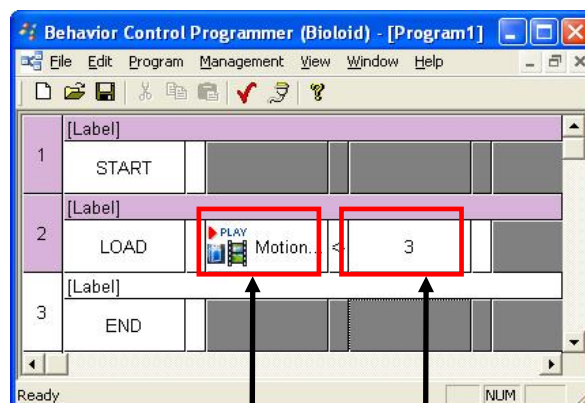
Page name	Forward walk
Page number	3
Data address	0x0000E600
Play count	1
Number of pose	4
Motion speed	32
Accel time	32
Next page No.	3
Exit page No.	0

If there are more than two motion pages, the motion is infinitely repeated when the next page is set as the first page from the last page.

Page name	Forward walk
Page number	3
Data address	0x0000E600
Play count	1
Number of pose	7
Motion speed	32
Accel time	32
Next page No.	4
Exit page No.	4

Page name	Forward walk2
Page number	4
Data address	0x0000E600
Play count	1
Number of pose	3
Motion speed	32
Accel time	32
Next page No.	3
Exit page No.	1

When the first page is assigned to the robot motion in the behavior control program, infinite repetition motion will be executed.

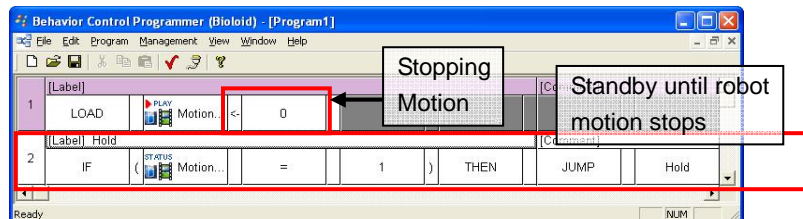


Robot Motion Icon

First Page Number of Infinite Repetition Motion

**Stopping Infinite** When “0” is input into the robot motion, the infinite repetition motion will

**Repetition Motion** stop and move to the exit page. When the exit page is “0”, it will execute all pages and stop in the last pose state.



**Example 20** Programming a walking motion of walking droid that moves backward when an obstacle is detected while it moves forward.

[ Step 1 ] Assemble a walking droid referring to the Quick Start.

[ Step 2 ] Run the motion program and program the following two walking motions.

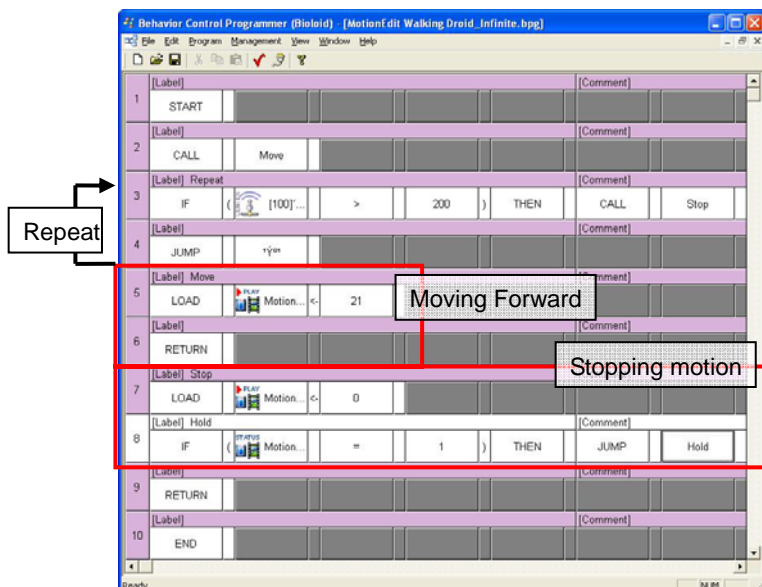
Page Number 12 : Moving forward motion

Page Number 13 : Moving backward motion

※ You can use a program already made for the above motions.

Download the same from 'C:\Program Files\Robotis \Boiloid\Applied Robots\Beginner\Walking Droid \DemoExample(Walking Droid).mtn'.

[ Step 3 ] Make a behavior control program like below to perform the above motions.



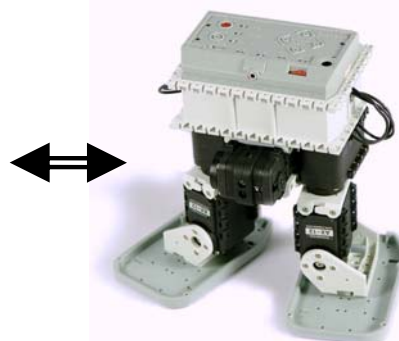
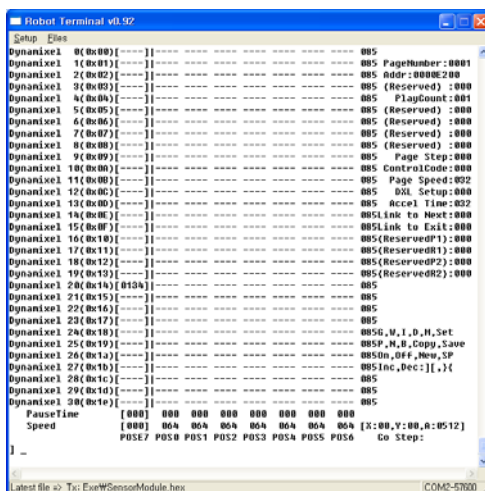
[ Step 4 ] Download the behavior control program to the robot and run the PLAY mode on the CM-5.



## 5. Robot Terminal

Once you get familiar with the robot you will want to edit motion in the text interface instead of the graphic user interface; being able to see all the information at once can be helpful. In this chapter we will learn how to edit motion using the robot terminal program.

**Robot Terminal** The Robot Terminal is a program that connects the CM-2+ to the PC. The CM-2+ does not have a screen or a keyboard, but the Robot Terminal program will allow you to input and output information using your PC. The information outputted from the CM-2+ will go to the PC through the serial cable and then print on screen through the Robot Terminal. Also, information inputted in the Robot Terminal via the keyboard will be sent to the CM-2+ through the serial cable.

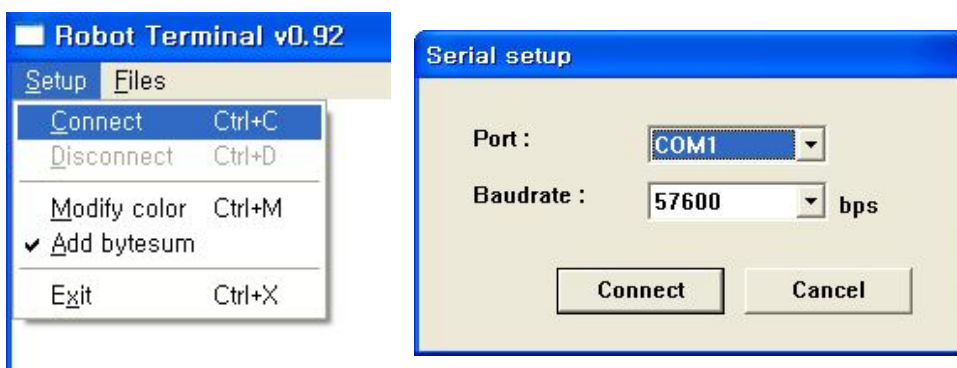


Keyboard

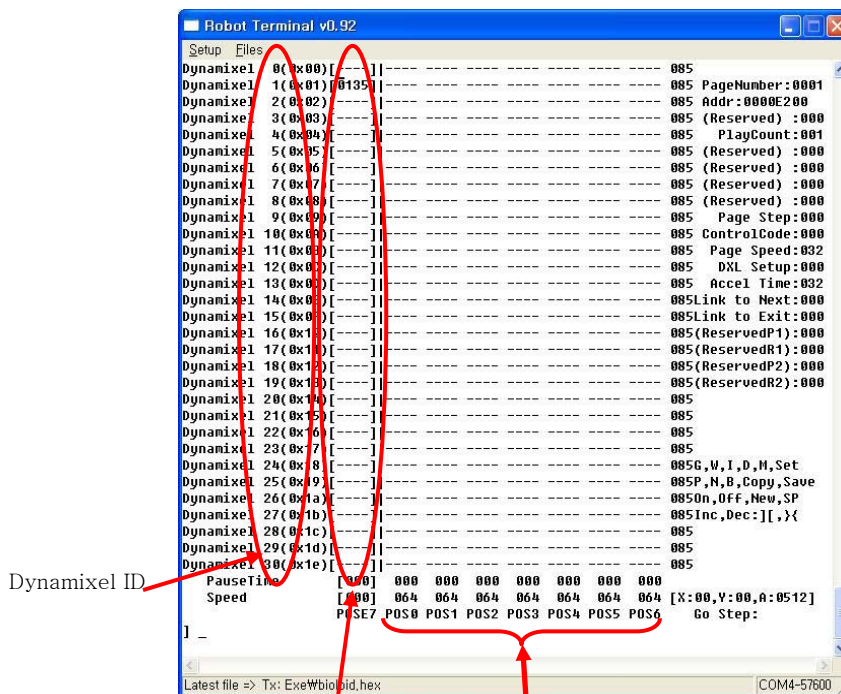
Screen

## Setting the Comport

If the connection fails after starting the Robot Terminal program, go to the setup menu and select Connect to set up the Com port, as shown in the figure below. Set it to the appropriate Com port and set the communication speed to 57600 bps. You only have to set the Com port once since this information will be saved inside the program.



Apply power to the CM-2+ and run the program mode. To do this, go into program mode by pressing the MODE button and then press the START button. The following screen should appear.



The position of the currently connected motors 7 poses

The first column on the left is the ID of the Dynamixel motor. The robot profile information is not shown here. The next column (which we call POSE 7) shows the values of the current angle positions of the Dynamixel units. From the numbers in POSE7 you can see that only one Dynamixel of ID 1 is connected.

The figure above shows the screen for editing a single motion page. A motion page is made of 7 poses and 64 bytes of page information. The size of one page is 512 bytes.

## Pose

The pose is a snap shot or instance of a motion. For example, in the figure below, you will need 5 poses to make the robot take one step to the side. A motion connects these poses smoothly.



**Command**

The following commands are available.

- Commands related to creating poses: ON, OFF, WRITE, SET, STEP, PLAY, GO, INSERT, MOVE, NAME, SAVE
- Commands related to editing pages: PAGE, BEFORE, NEXT, COPY, NEW

A multi-jointed robot motion can be edited using these commands. Let's take a look at each one.

**Off**

This will turn off the torque of the joint. If you input OFF, the joint angle value for POSE7 will disappear (see figure on the right). You can now move this joint by hand.

**On**

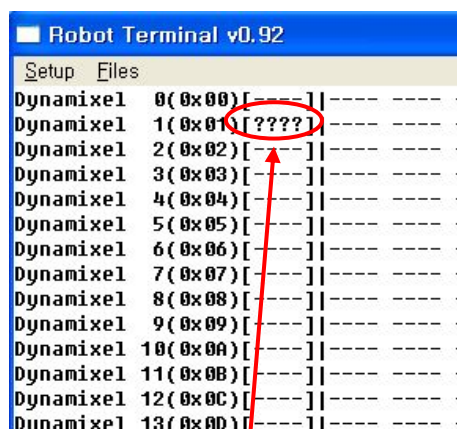
After moving the joint to the desired position, type the ON command to see the joint angle value at POSE7. The torque will be back on, locking its joint position.

**TIP**

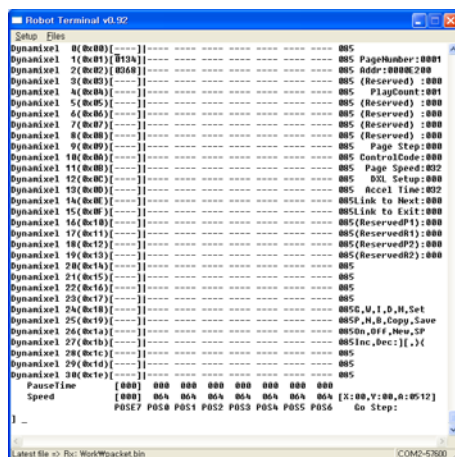
You can list several Dynamixel IDs after an OFF or ON command to turn them on or off all at once.

**Write**

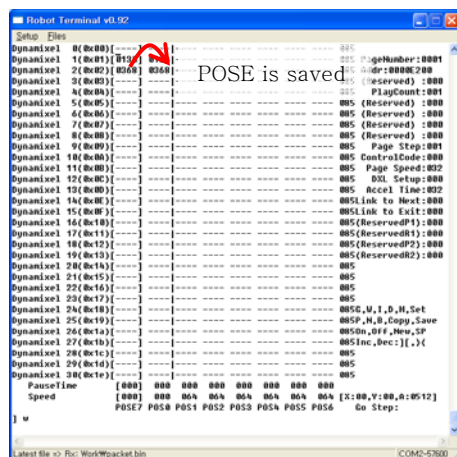
After setting the joint angles to the desired positions, type in the WRITE command. The joint angle values of the POSE7 will be added to the pose. Let's make several more poses this way.



Torque Off state with OFF instruction



After executing the  
ON command



After executing the  
WRITE command

**Play**

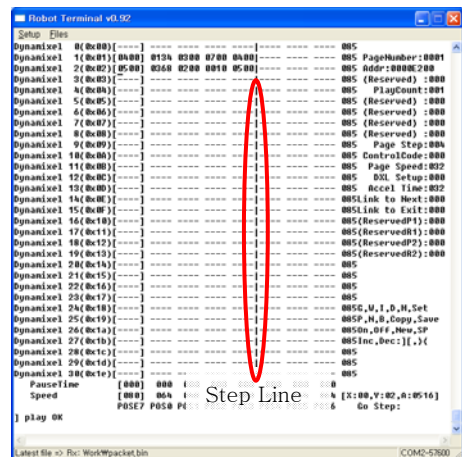
We will now check the motion that connects the inputted poses. If you type in “play,” the motion will be played. The poses from POSE0 to the last inputted pose will be played. You can see where the last inputted pose is by the Step line.

**TIP**

You can play another motion page by typing in “Play [page number].” For example, “Play 3” will play the motion of page number 3.

**Go**

After playing a motion, sometimes you will want to edit it. Here, you can use the “go” command. This command will take you to a certain pose. For example, “go 1” will make the robot move to the configuration of pose1 and the joint angle values of POSE1 will be copied into POSE7. The Dynamixel motors will move at a constant speed.



[Playing from Pose 0 to Pose 3]

**Write [pose number]**

After using the GO command and editing the pose data using the OFF, ON commands, you will want to save the new joint values of POSE7. This can be done by typing in “Write [pose number].” And the joint values for POSE7 will be saved in the specified “pose number.”

**Insert**

While editing, you will sometimes want to place the current pose (POSE7) between two other poses. To do this, use the “insert” command. The format is “insert [pose number].”

**Delete**

A pose can be deleted using the “delete” command. Typing in “delete” without a parameter will move the step line up one column. If you want to delete a certain pose then type in “Delete [number].”

**Step**

Sometimes during motion editing you will want to change the location of the Step line. For example, say that you made 4 poses, but you want to run only the first two poses. Typing in “step 2” will move the step line to the beginning of pose 2 and only POSE0 and POSE1 will be played.

<b>Name</b>	With this function you can give a name to a page. This will be useful later.
<b>Save</b>	During editing, the motion data is stored inside the CM-2+'s RAM. When you are finished creating the motion, use the save command to save the motion page to the flash memory.
<b>Page</b>	Type in "page [page number]" to jump to another page.
<b>Before</b>	Moves to the previous page.
<b>Next</b>	Moves to the next page.
<b>Note.</b>	Make sure to save the motion data before moving to another page since it will be deleted if not saved.
<b>Copy</b>	To copy the data of a certain page onto the current page, type in Copy [page number]." The copied data is not saved in the flash memory yet.
<b>New</b>	This command will erase all the information inputted on the current page.

Next we will be learning about editing the page information. The page information is located on the upper right corner of the screen. The following are names of the items and are not commands. They can be edited using the "Set" command.

<b>Speed</b>	Even though the joint values have been inputted properly, you might still not be able to get the desired motion unless you set the speed between poses correctly. You can do this by setting the value of the speed at the bottom of the screen. The following shows how this is done.
--------------	--

Move the cursor to the item that you want to set.  
Press the "]" or "[" key to increase or decrease the value of the item.  
Press the "{" or "}" key to change the magnitude.  
By typing in "Set [value]" the value can be set at once.

The method above can be used to set not only the speed, but also all the other items below.

```

Robot Terminal v0.92
Setup Files
Dynamixel 0(0x00)[----]----- 085
Dynamixel 1(0x01)[0134] 0134 085 PageNumber:0001
Dynamixel 2(0x02)[0368] 0368 085 Address:0000200
Dynamixel 3(0x03)[----]----- 085 (Reserved):000
Dynamixel 4(0x04)[----]----- 085 PlayCount:00
Dynamixel 5(0x05)[----]----- 085 (Reserved):000
Dynamixel 6(0x06)[----]----- 085 (Reserved):000
Dynamixel 7(0x07)[----]----- 085 (Reserved):000
Dynamixel 8(0x08)[----]----- 085 (Reserved):000
Dynamixel 9(0x09)[----]----- 085 Page Step:001
Dynamixel 10(0x0A)[----]----- 085 ControlCode:00
Dynamixel 11(0x0B)[----]----- 085 Page Speed:00
Dynamixel 12(0x0C)[----]----- 085 DKL Setup:000
Dynamixel 13(0x0D)[----]----- 085 Accel Time:032
Dynamixel 14(0x0E)[----]----- 085Link to Next:000
Dynamixel 15(0x0F)[----]----- 085Link to Exit:000
Dynamixel 16(0x10)[----]----- 085(ReservedP1):000
Dynamixel 17(0x11)[----]----- 085(ReservedR1):000
Dynamixel 18(0x12)[----]----- 085(ReservedP2):000
Dynamixel 19(0x13)[----]----- 085(ReservedR2):000
Dynamixel 20(0x14)[----]----- 085
Dynamixel 21(0x15)[----]----- 085
Dynamixel 22(0x16)[----]----- 085
Dynamixel 23(0x17)[----]----- 085
Dynamixel 24(0x18)[----]----- 085G,W,I,D,M,Set
Dynamixel 25(0x19)[----]----- 085P,N,B,Copy,Save
Dynamixel 26(0x1A)[----]----- 085On,Off,New,SP
Dynamixel 27(0x1B)[----]----- 085Inc,Dec:][,}{
Dynamixel 28(0x1C)[----]----- 085
Dynamixel 29(0x1D)[----]----- 085
Dynamixel 30(0x1E)[----]----- 085
PauseTime [000] 000 000 000 000 000 000 000
Speed [000] 000 064 064 064 064 064 064 [A:00,V:00,A:0512]
POSE7 POSE8 POSE1 POSE2 POSE3 POSE4 POSE5 POSE6 Go Step:
] w
Pause time and Speed
Latest file => F:\Work\packet.bin COM2-57600

```

Page  
information

## PauseTime

Sometimes you will want to pause for a short time after a pose. For example, you might want the robot to pause for a while after it has finished doing a bowing motion. If you give it a “pause time” value, it will pause for 7.8 msec per the set value and then move on to the next pose. The figure above shows an example of the robot pausing for 0.5 s (40\*7.8 msec) after POSE1 is played.

- Accel. Time** Every time a motion is played, the joints go through a process of constant acceleration and deceleration. The acceleration time is the time of acceleration and deceleration (ramp up time plus ramp down time). Reducing the acceleration time will make the motors accelerate faster and stress the joints. Increasing the acceleration time might create an interval that makes it impossible to complete a motion.
- Page Speed** Use this when you want to adjust the play speed of the whole page. The default value is 32. Setting the value to 64 would be the same as doubling the speed of each of the poses individually.
- Link to Next** You can set the next motion page to be executed after the current page is finished being played. This can be done by inputting the page to be played next in the “Link to Next” item. If you don't need to play another motion page then set the value to 0 (default value is 0). If you set the value for “Link to Next” as the number of the current page, then playback will continue infinitely in a loop. In this case, you can input the escape key stop the playback. Playback will be stopped after the Link to Exit motion is completed.
- Link to Exit** During motion play, if a signal is received, playback will end after executing the page defined by this value. If this feature is not needed, then set its value to 0.

### Other Things to Keep in Mind

- Page 0, 1** Page 0 and page 1 have special functions, so it is recommended that you do not use them.

#### “Are you sure?”

The “Are you sure?” message will appear in the following cases.

- When you move to another page without saving.
- When you use the “go” command with a pose that is outside of the step line.
- When you use the “new” command.

- Abbreviation** You can use only the first letter of the commands that are used often.



## 6. Wireless Remote Control

Robots that use the CM-2+ support two types of wireless communication. With these methods, you can control the robots remotely or allow the robots to send and receive data between each other.

The first method involves sending data using the IR (infrared rays) transmitter-receiver function of the AX-S1. The second by attaching the Zigbee module ZIG-100, a dedicated robot wireless device, to the CM-2+. With this attachment, the robot can communicate via the RF method.



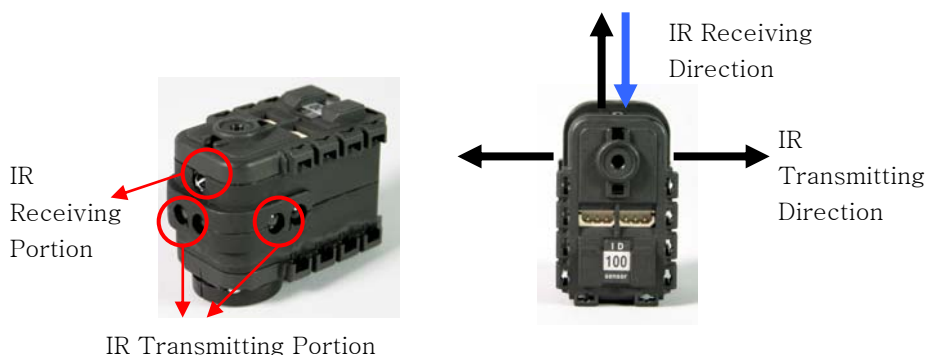
AX-S1 : Infrared (IR)  
Communication



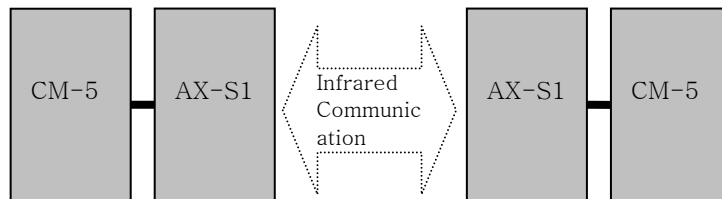
ZIG-100 : RF Communication

### 6-1. Infrared Wireless Manipulation

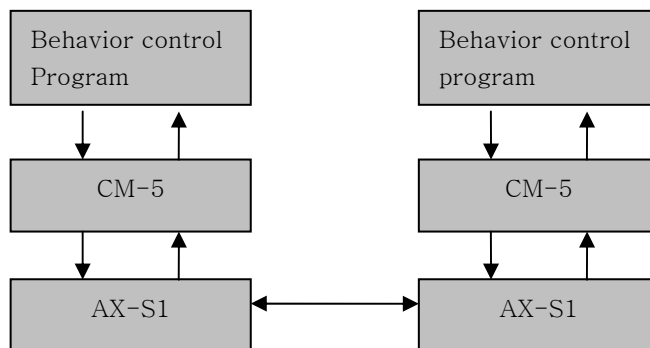
**Using Ax-S1 (IR Communication)** AX-S1 contains a transmitting/receiving portion allowing IR communication. As in the following figure, the AX-S1 transmits data in three directions and receives data in the front. Therefore, you have to be careful to adjust the direction of the AX-S1 for IR wireless communication.



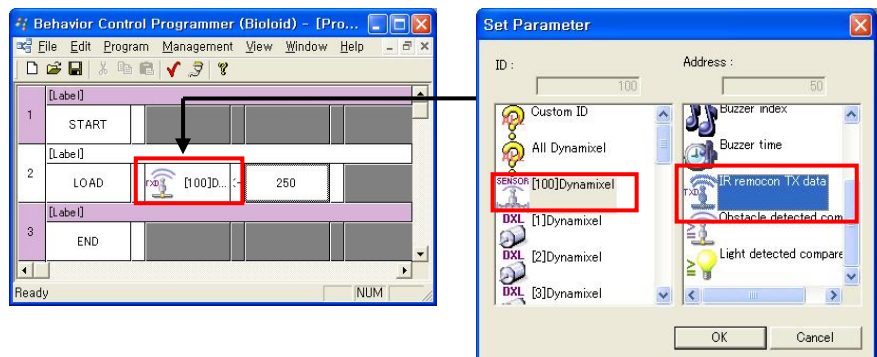
**IR Communication H/W** For the infrared communication, more than two AX-S1s and CM-2+s are needed.



**IR Communication S/W** To control the Bioloid using infrared communication, a behavior control program for communication is needed.

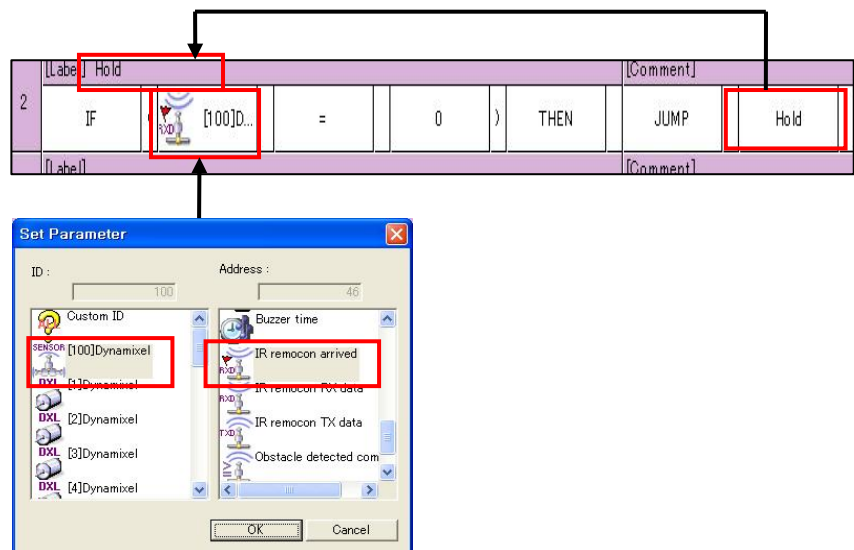


**Behavior Control Program ( for Transmitting )** A behavior control program for transmitting data through wireless connection should be downloaded in the CM-2+. A command for transmitting is 'IR remocon TX data'. When you execute a command composed as in the following figure, a value corresponding to the inputted number or variable. [The range of transmittable value is 0~65535.]

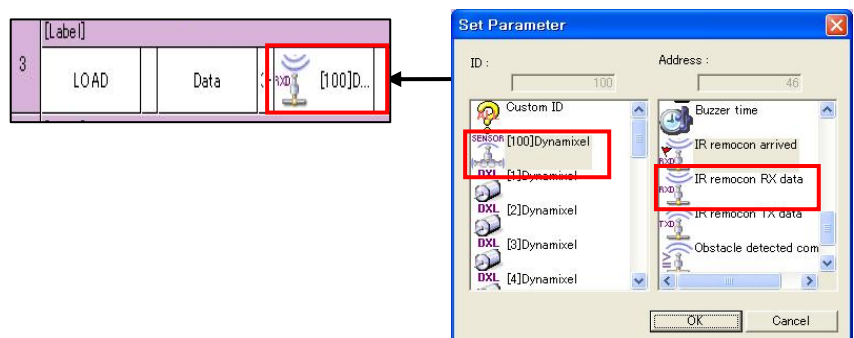


**Behavior Control Program [for Receiving]** Transmission of data can be performed whenever needed, but as to receipt, it is hard to know when the data arrives. Therefore, a behavior control program is more complicated than that of transmission.

To receive data that you do not know when it arrives, a command of standby for receiving is needed. 'IR remocon arrived' becomes 1 when there is an incoming data and 0 when there is none. You can make up a standby command as follows.



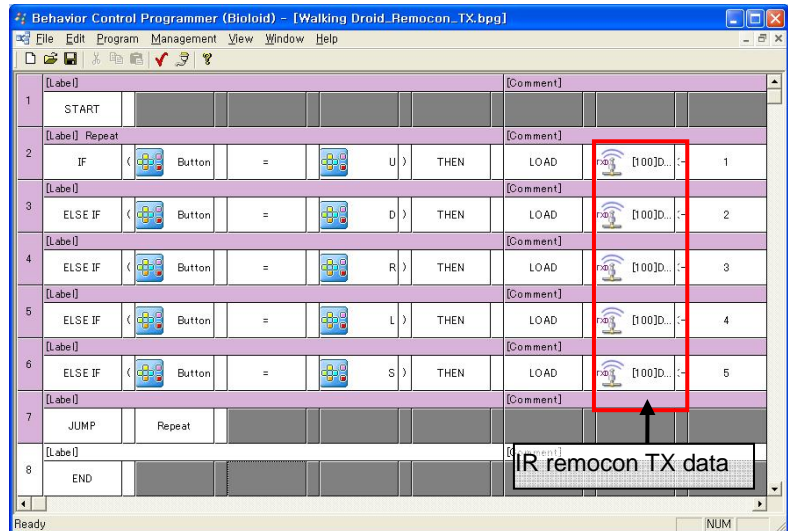
When data has arrived, the standby command stops and the data is read by 'IR remocon RX data'. Create a command for the robot to receive data as shown in the following figure.



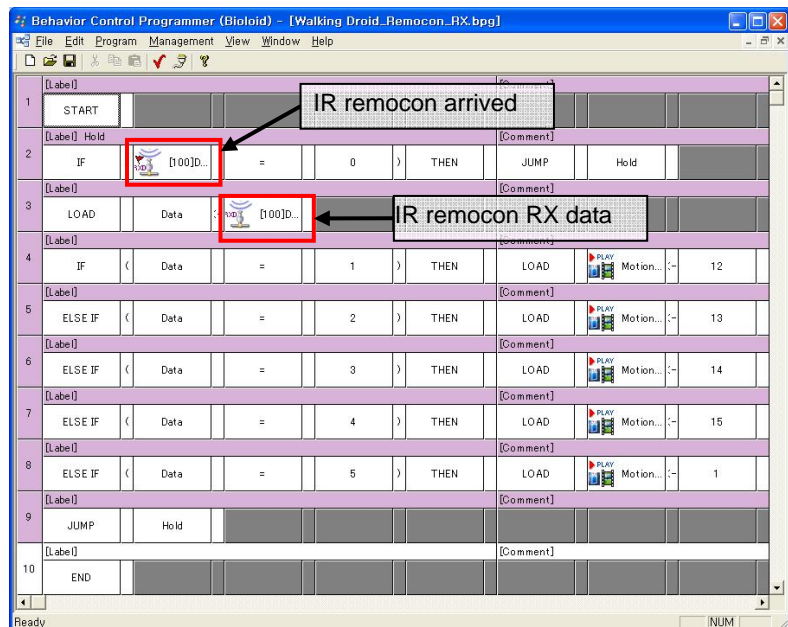
**Example 21** Manipulates the walking droid of Example 18. by wireless

[ Step 1 ] Run Example 18.

[ Step 2 ] Make a behavior control program for transmitting data.



[ Step 3 ] Make a behavior control program for receiving data.

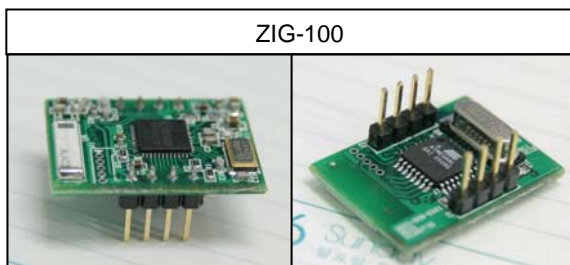


[ Step 4 ] Download the behavior control program for transmitting data in the remote controller and the behavior control program for receiving data in the robot, Then, execute the play mode of each CM-5. Verify that the walking droid can be controlled wirelessly by pressing the direction buttons on the CM-5.

## 6-2. ZIGBEE Wireless Manipulation

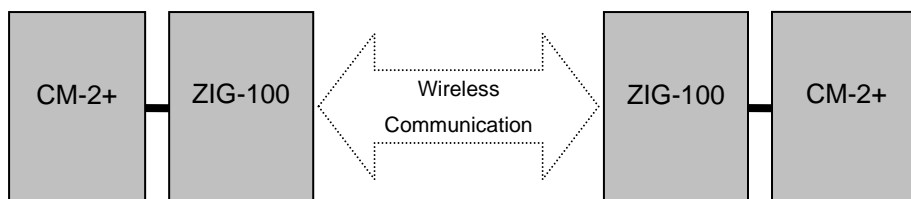
### Zigbee

ZIG-100 is a device used for wireless communication of Bioloid. Zigbee is a technique for communication frequently used for PAN[Personal Area Network] like Bluetooth.



### CM-2+ and ZIG-100

For the communication between the Zigbee and Bioloid, the ZIG-100 should be mounted in the CM-2+. You have to disassemble the CM-2+ and insert ZIG-100 module in a place of ZIGBEE on the circuit board. To transmit and receive data, more than two sets of CM-2+ s and ZIG-100 are required.



CM-2+ (Before ZIG-100 is mounted)



CM-2+ (After ZIG-100 is mounted)

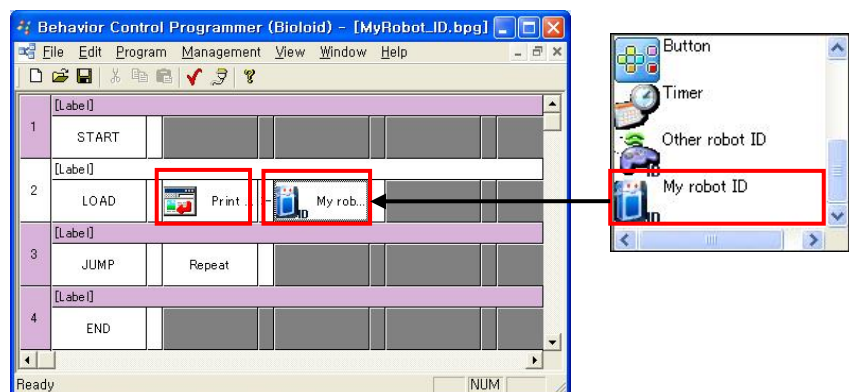
**Zigbee ID**

Each the ZIG-100 has a proper ID. Therefore, for the communication between the ZIG-100s, they should know each other's ID. Once the IDs are known, one-to-one communication between the ZIG-100s is possible. When the ZIG-100 is used for a special purpose, a broadcasting mode allowing the ZIG-100 to communicate with all other surrounding ZIG-100s is also possible. The details are described in the ZIG-100 manual.

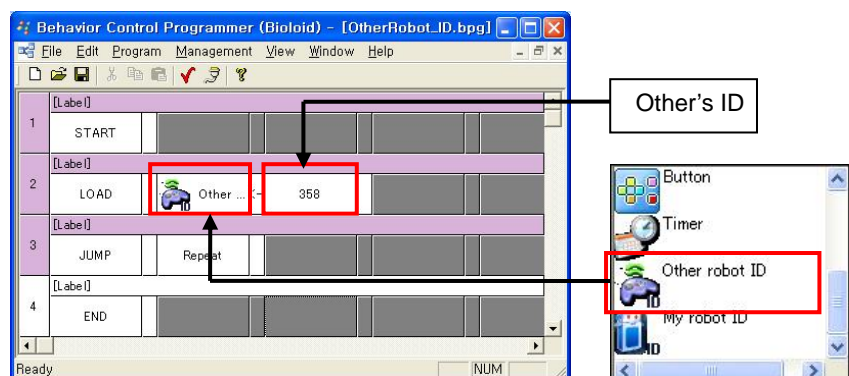
**Setting ID**

To assign an ID of one ZIG-100 to another ZIG-100, the behavior control program is used. The following figure explains the method in assigning an ID for other ZIG-100s. When you create a behavior control program, download and run the program in the CM-2+ where the ID is assigned. Please refer to 'C:\WProgram Files\Robotis\Bioloid\Example\Example[RF Verifying My ID].bpg' and 'Example [RF Assigning Other's ID].bpg'.

Behavior control program to verify an ID of itself

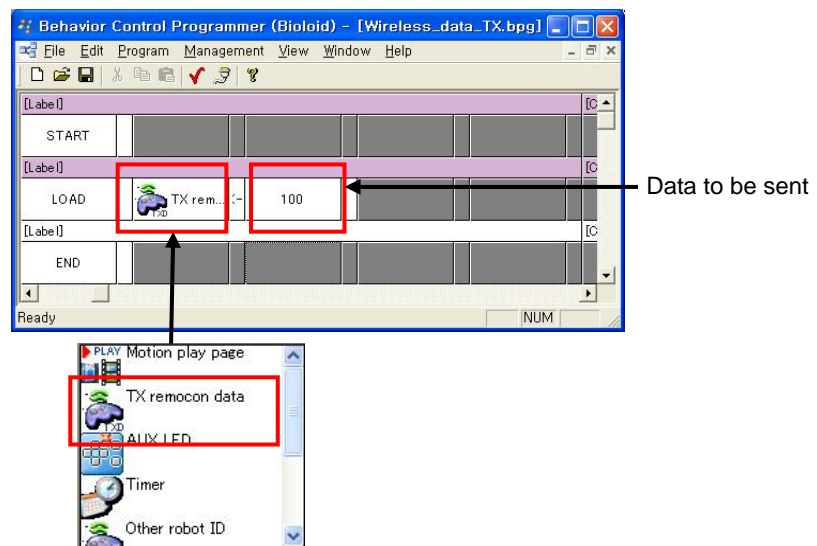


Behavior control program to assign an ID for the other

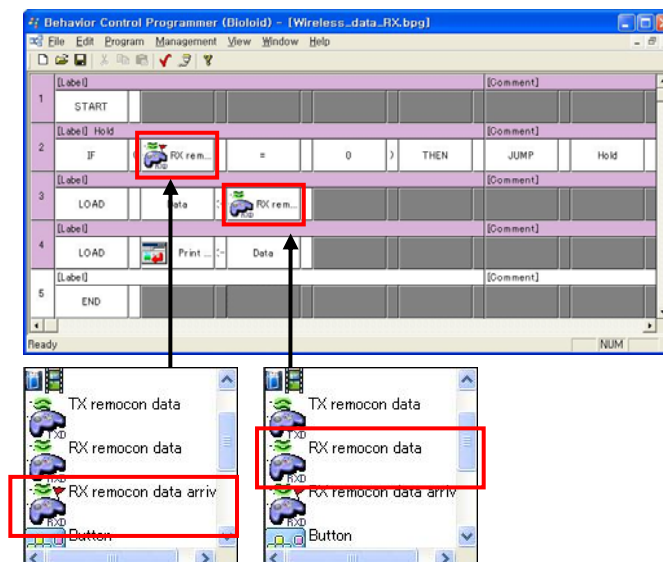


**ZIG-100 Communication S/W** Zig-100 also uses the behavior control program as S/W. The behavior control program has the same structure as that of the infrared wireless communication but different commands. The following figure shows a behavior control program for transmitting and receiving data.

**Behavior Control Program [for Transmitting]** 'CM-2+ > TX romocon data' is used as a command. The method is similar to that of the infrared communication.



**Behavior Control Program [for Receiving]** 'CM-2+ > RX romocon data' is used as a command. The method is similar to that of infrared communication.



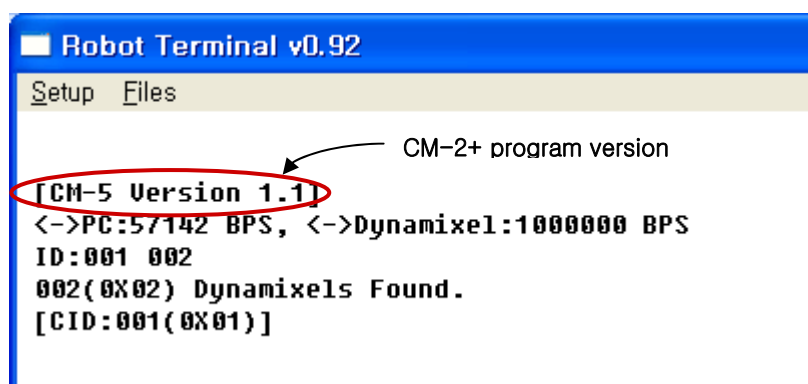
## 7. Management Mode

This chapter explains about using Management Mode. In “manage mode” you can check the robot status and check or change the Dynamixel settings.

### 7-1. SETTING THE ID AND DYNAMIXEL SEARCH

**Robot Terminal** Run the Robot Terminal program. In the previous chapter, we explained how the CM-2+ and the PC are connected. Since we explained comport setup in detail in previous chapter, we will omit it here.

**Initial State** When you execute the “manage mode” the following screen will show up in the Robot Terminal.



Here you can see the number and the IDs of the connected Dynamixel units. If what appears on the screen is different from the actual configuration, check the following.

- Do all the Dynamixel units have different IDs?
- Do the communication speeds between the Dynamixel units and the CM-2+ agree?
- Are the cables connected properly and securely?

To check the wiring, turn the power off on the CM-2+ and then turn it back on. Check if the LEDs on the Dynamixel units are blinking. If not, check the wiring again.



If the wires are connected properly you can check the communication speed and find the IDs by using the SEARCH command. This will be covered later.

#### Command Format

Type in a command followed by a number (parameter). The following are some examples.

- ID 10
- Dump
- WR 10, 1
- RD 10, 2

#### HELP

Type in help to see the available functions in manage mode.

```

[CID:001(0X01)] help

DUMP(D) : Dump control table.
ID [ID_NUM] : Set ID of dynamixel.
CID [ID_NUM] : Change Control ID.
READ [ADDR][LEN] : Read data. ex)Read 10 2 (Read from 0x10, length 2)
WRITE [ADDR][DATA1].. : Write data. ex)Write 14,1 (Write 1 at 0x14)
REG_WR [ADDR][DATA1].. : Register write instruction
ACTION : Action REG_WR instruction
Go [POSITION][SPEED]: Goto the position with the speed.
HEX [NUM][NUM]... : Transmite raw data. ex)Hex FF FF 01 03
RESET : Dynamixel Reset.
PING [NUM]: ex) Ping NUM ID dynamixel.
SWR [ADDR][LEN][ID][DATA]...[ID][DATA]... : Sync Write.
SCAN [NUM] : SCAN linked dynamixel in 0~NUM in current baud rate.
LED [NUM] : Blink LED of NUM ID. 'B','N' for ID change. 'Q' for Quit.
BAUD [NUM] : Set baud rate ex) BAUD 22(57600BPS), BAUD 1(1MBPS)
SEARCH : Search ID and baudrate of all linked dynamixels.
Update [START_ID] [END_ID] : Dynamixel firmware update.(system user only)

```

Copyright ROBOTIS CO.,LTD.

```
[CID:001(0X01)] _
```

## CID

CID is the abbreviation of the Control ID. This shows the ID of the Dynamixel that the CM-2+ is controlling. CID is also used as a prompt character in manage mode.

```

[CM-5 Version 1.1]
<->PC:57142 BPS, <->Dynamixel:1000000 BPS
ID:001 002
002(0X02) Dynamixels Found.

```

**[CID:001(0X01)]** → Indicates the control of Dynamixel that has the ID of 0x01

To change the ID of the Dynamixel that the CM-2+ controls to number 3, type in the following commands.

```
[CID:001(0X01)]
[CID:001(0X01)] cid 3
[CID:003(0X03)]
[CID:003(0X03)] _
```

After the command above, only the Dynamixel with an ID of 0x03 will react. Communication between the CM-2+ and Dynamixel will not occur even if you run the CID command.

**ID** Use the ID command when you want to change the IDs of all the connected Dynamixel units.

- Usage: ID [ID number]
- Example) ID 2 Set the connected Dynamixel units' ID to 2.  
The ID command will change the IDs of all the connected Dynamixel units regardless of the value of the CID. Therefore, when you use the ID command, make sure that there is only one Dynamixel connected to the CM-2+.

When you use the ID command make sure that there is only one Dynamixel connected to the CM-2+

**ID 254** Using ID number 254 will send commands to all Dynamixel units. A Dynamixel will only react to a command with its own ID or with ID 254, but it will not send back a packet for commands sent with ID 254. ID 254 is also called the "Broadcasting ID."

**SCAN** You can find the IDs of the Dynamixel units connected to the CM-2+ by running the SCAN command. If you type in SCAN N, the program will scan the Dynamixel units from number 0 to N. The SCAN command will only work if the communication speed between the CM-2+ and the Dynamixels is set properly.

- Usage: Scan [number of IDs]

```
[CID:003(0X03)] scan 10
[000:-][001:0][002:0][003:-][004:-][005:-][006:-][007:-][008:-][009:-]
002(0X02) Dynamixels Found.
[CID:003(0X03)] _
```

**SEARCH**

If you are not sure if the communication speed between the CM-2+ and the Dynamixel is set properly, you can use the SEARCH command to search the Dynamixel units.

```
[CID:003(0X03)] search  
Found! BAUD_REG:001, ID:001  
Found! BAUD_REG:001, ID:002
```

```
[CID:003(0X03)] _
```

The SEARCH command is slow and it could find duplicates if similar baud rates are used. This is because UART communication is somewhat robust against baud rate error.

**LED**

Sometimes you will want to check the ID of each Dynamixel unit connected to the CM-2+ . Type in LED ID and the LED of the Dynamixel unit of the selected ID will blink. Type B and N to change the ID. Typing Q will end the LED command.

```

[CID:003(0X03)] led 1
B(before),N(next),Q(Quit)
LED Blink ID:001
LED Blink ID:002
LED Blink ID:003
LED Blink ID:004
LED Blink ID:004
[CID:003(0X03)]

```

When N is typed in

When Q is typed in

## 7-2. Other Commands

### READ

This command is used to read the data values in the control table of a Dynamixel unit.

The READ command is used as the following.

- Usage: READ [ADDRESS] [Data Length for Reading]

The example below shows a command that reads 1 byte from Address 25 of the Dynamixel with an ID of 1.

```

[CID:002(0X02)] rd 25 1
->[Dynamixel]:255 255 002 004 002 025 001 221 LEN:008(0X08)
<-[Dynamixel]:255 255 002 003 000 000 250 LEN:007(0X07)
[CID:002(0X02)]

```

### WRITE

This command is used to change a data value of the control table.

The WRITE command format is as the follows.

- Usage: WRITE [Address] [Data] [Data] [Data]...

In the example below, you can verify that the LED turns on and off when 1 and 0 is written to Address 25.

```

[CID:002(0X02)] wr 25 1
->[Dynamixel]:255 255 002 004 003 025 001 220 LEN:008(0X08)
<-[Dynamixel]:255 255 002 002 000 251 LEN:006(0X06)
[CID:002(0X02)] wr 25 0
->[Dynamixel]:255 255 002 004 003 025 000 221 LEN:008(0X08)
<-[Dynamixel]:255 255 002 002 000 251 LEN:006(0X06)
[CID:002(0X02)]

```

**Dump**

This shows the control table values of a Dynamixel unit. The following shows the information that is dumped. Refer to the motor manual for more information about control tables.

```

Setup  Files

[CID:001(0X01)] dump
->[Dynamixel]:255 255 001 004 002 000 058 190 LEN:008(0X08)
<-[Dynamixel]:255 255 001 060 000 012 000 017 001 001 000 000 000 255 003 131 0
85 060 140 255 003 002 004 004 000 040 000 205 003 001 000 001 001 032 032 143 0
01 000 000 255 003 142 001 000 000 000 000 091 037 000 000 000 000 032 000 000 0
00 149 001 000 000 000 000 099 LEN:064(0X40)

[EEPROM AREA]
MODEL_NUMBER_L      (R) [000(0X00)]:012(0X0C)
MODEL_NUMBER_H      (R) [001(0X01)]:000(0X00)
VERSION              (R) [002(0X02)]:017(0X11)
ID                   (R/W)[003(0X03)]:001(0X01)
BAUD_RATE             (R/W)[004(0X04)]:001(0X01)
RETURN_DELAY_TIME     (R/W)[005(0X05)]:000(0X00)
CW_ANGLE_LIMIT_L      (R/W)[006(0X06)]:000(0X00)
CW_ANGLE_LIMIT_H      (R/W)[007(0X07)]:000(0X00)
CCW_ANGLE_LIMIT_L     (R/W)[008(0X08)]:255(0XFF)
CCW_ANGLE_LIMIT_H     (R/W)[009(0X09)]:003(0X03)
(RESERVED)            (R/W)[010(0X0A)]:131(0X83)
LIMIT_TEMPERATURE     (R/W)[011(0X0B)]:085(0X55)
DOWN_LIMIT_VOLTAGE    (R/W)[012(0X0C)]:060(0X3C)
UP_LIMIT_VOLTAGE      (R/W)[013(0X0D)]:140(0X8C)
MAX_TORQUE_L          (R/W)[014(0X0E)]:255(0XFF)
MAX_TORQUE_H          (R/W)[015(0X0F)]:003(0X03)
RETURN_LEVEL          (R/W)[016(0X10)]:002(0X02)
ALARM_LED             (R/W)[017(0X11)]:004(0X04)
ALARM_SHUTDOWN        (R/W)[018(0X12)]:004(0X04)
(RESERVED)            (R/W)[019(0X13)]:000(0X00)
DOWN_CALIBRATION_L    (R/W)[020(0X14)]:040(0X28)
DOWN_CALIBRATION_H    (R/W)[021(0X15)]:000(0X00)
UP_CALIBRATION_L      (R/W)[022(0X16)]:205(0XCD)
UP_CALIBRATION_H      (R/W)[023(0X17)]:003(0X03)
Any Key to Continue...

```

Press any key to continue dump.

**GO**

This command moves the Dynamixel unit to the specified position. The GO command is used like the following.

- Usage: GO [Position Value] [Speed Value]

Here, the range of parameter values is from 0 to 1023. If you take a look at the packet, you can see that the WRITE command has been executed starting from Address 30, which corresponds to goal position and goal speed.

```
[CID:001(0X01)] go 100 80
->[Dynamixel]:255 255 001 007 003 030 100 000 000 000 034 LEN:011(0X0B)
<-[Dynamixel]:255 255 001 002 000 252 LEN:006(0X06)
[CID:001(0X01)] _
```

## PING

This command does not execute any special tasks, but is used to check to see if a Dynamixel is connected. The Dynamixel will return a packet even when it receives a Broadcasting ID with this command.

- Usage: PING [ID]

```
[CID:001(0X01)] ping 1
->[Dynamixel]:255 255 001 002 001 251 LEN:006(0X06)
<-[Dynamixel]:255 255 001 002 000 252 LEN:006(0X06)
[CID:001(0X01)] _
```

## REG\_WR

This command registers the command WRITE. The command is only registered; not executed. The format is the same as the WRITE command. But it will only execute when the ACTION command is given.

## ACTION

This command executes the WRITE command that is registered by REG\_WR. The example below shows the process of turning on a LED using the REG\_WR command. The LED will actually be turned on with the Action command.

```
[CID:001(0X01)] reg_wr 25 1
->[Dynamixel]:255 255 001 004 004 025 001 220 LEN:008(0X08)
<-[Dynamixel]:255 255 001 002 000 252 LEN:006(0X06)
[CID:001(0X01)] action
->[Dynamixel]:255 255 254 002 005 250 LEN:006(0X06)
<-[Dynamixel]:
No Data[at Broadcast ID 0xFE] LEN:000(0X00)
[CID:001(0X01)]
```

The Action command is executed with the Broadcasting ID. The REG\_WR and Action commands are useful when you want to actuate several Dynamixels starting at the same time.

**SYNC\_WR**

When you want to write to several Dynamixel units and if the Write Addresses are all the same, you can use the SYNC\_WR command to write to all of the Dynamixels at once. The format of a SYNC\_WR command is as follows.

Usage: SWR [ADDRESS] [LENGTH] [ID] [DATA0] [DATA1] ...[ID] [DATA0] [DATA1]...

The following example shows how to move a Dynamixel of ID = 0 to position 512(0x200) at a speed of 80(0x80) and a Dynamixel of ID = 1 to position 272(0x110) at a speed of 80(0x80). SYNC\_WR is a broadcasting command.

```

[CID:001(0X01)] swr 30 4 0 0 2 80 0 1 0 1 80 0 000 000 002 080 000 001 000 001 080 0
->[Dynamixel]:255 255 254 014 131 030 004 000 000 002 080 000 001 000 001 080 0
00 170 LEN:018(0X12)
<-[Dynamixel]:
No Data[at Broadcast ID 0xFE] LEN:000(0X00)
[CID:001(0X01)] _

```

**Baud**

This command is used to change the baud rate of the CM-2+ and Dynamixel controlling UART. The baud rate is calculated using the following equation.

$$\text{Speed(BPS)} = 2000000 / (\text{Parameter Value} + 1)$$



## Parameter Values for Important Baud Rates

Parameter	Set BPS	Goal BPS	Error
1	1000000.0	1000000.0	0.000%
3	500000.0	500000.0	0.000%
4	400000.0	400000.0	0.000%
7	250000.0	250000.0	0.000%
9	200000.0	200000.0	0.000%
16	117647.1	115200.0	-2.124%
34	57142.9	57600.0	0.794%
103	19230.8	19200.0	-0.160%
207	9615.4	9600.0	-0.160%

The Baud command changes the baud rate of the CM-2+ itself and all the Dynamixel units that are connected to the CM-2+.

Usage: BAUD [Calculated parameter value]

**Note**

A maximum Baud Rate error of 3% is within the tolerance of UART communication.

**RESET**

The RESET command will change all the settings of the Dynamixel unit back to the factory initial settings.

Usage: reset [ID]

Do not use the RESET command if possible.

```
[CID:001(0X01)] reset 1
Are you sure?(y/N)
->[Dynamixel]:255 255 001 002 006 246 LEN:006(0X06)
<-[Dynamixel]:255 255 001 002 000 252 LEN:006(0X06)
[CID:001(0X01)]
```

**H** The H command will send the numbers that are typed into the Robot Terminal as text to the Dynamixel units in binary format. The H command is useful when testing the packet communication protocol.

Usage: H [Parameter] [Parameter] [Parameter]...

So far we have learned the functions of several manage mode commands.

## 8. Information for Advanced Users

This chapter is for advanced users who have experience with microprocessors. In order to understand the following material you will need to have knowledge of hexadecimal, binary numbers, and ASCII code. Finally, we briefly explain how to control the CM-2+ using C language.

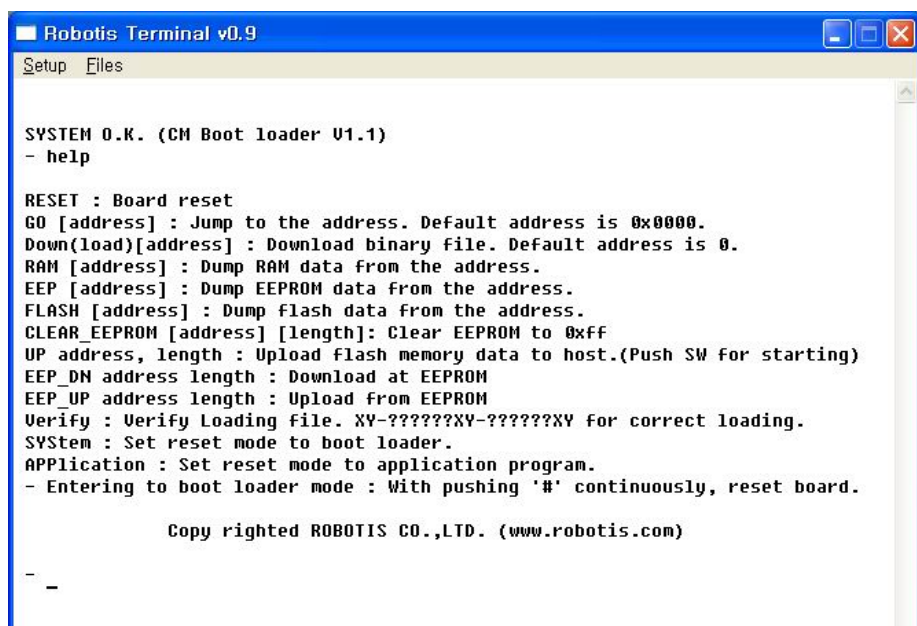
### 8-1. Boot Loader

**Boot Loader** When power is applied to the CM-2+ unit, the “CM Boot Loader” program in Reset Vector is executed. The “CM Boot Loader” program does not have as many functions as a PC operating system, but it has the following basic features: uploading and executing user created programs to the CM-2+ unit memory, verifying the data in the memory, and downloading programs back to a PC. All numbers are treated in hexadecimal.

**Caution** If you do not fully understand the system, do not use the Boot Loader.

**Execution** In the Robot Terminal, press and hold the # key and press the mode switch to go into the Boot Loader.

The figure below shows the Boot Loader screen. At this point, type Help” and press Enter, and the following message will appear.



```

Robotis Terminal v0.9
Setup Files

SYSTEM O.K. (CM Boot loader V1.1)
- help

RESET : Board reset
GO [address] : Jump to the address. Default address is 0x0000.
Down(load)[address] : Download binary file. Default address is 0.
RAM [address] : Dump RAM data from the address.
EEP [address] : Dump EEPROM data from the address.
FLASH [address] : Dump flash data from the address.
CLEAR EEPROM [address] [length]: Clear EEPROM to 0xff
UP address, length : Upload flash memory data to host.(Push SW for starting)
EEP_DN address length : Download at EEPROM
EEP_UP address length : Upload from EEPROM
Verify : Verify Loading file. XY-?????XY-?????XY for correct loading.
SYStem : Set reset mode to boot loader.
APPlication : Set reset mode to application program.
- Entering to boot loader mode : With pushing '#' continuously, reset board.

Copy righted ROBOTIS CO.,LTD. (www.robotis.com)
- _
  
```

This is a summary of the functions of the Boot Loader. Let's take a closer look at them, one at a time.

## Download

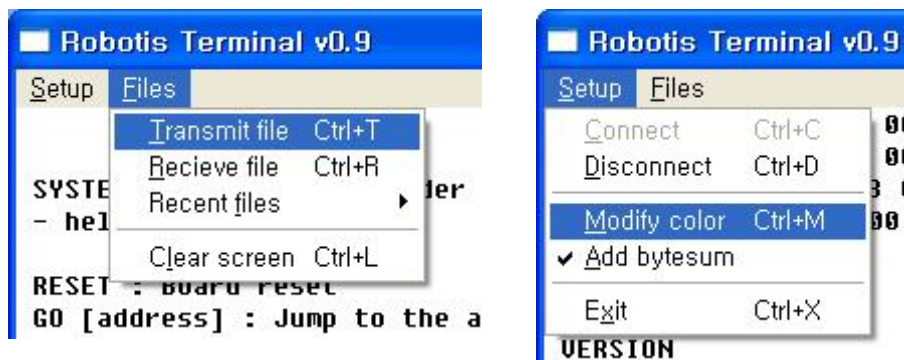
Let's learn how to download a provided Firmware or a program you have created onto the CM-2+ unit. Let's try downloading a program called CM-2+ .hex.

Type in the command "load." The following message should appear. This message indicates that the data is ready to be written to address 0.

```
- load
Write Address : 00000000
Ready..
```

Next, select "Transmit file" in the "Files" menu from the Robot Terminal program as shown below. It is recommended to check the "Add bytesum" menu item in the "Setup" menu as shown below.

Selecting a File for Transmission



Select "CM-2+ .hex" of the CD as the file to be transmitted. The selected file will be transmitted to the CM-2+ through the serial cable.

## Transmission Complete

When the transmission is finished, a "Checksum:xx-xx" message will appear on the screen as shown below. If the two numbers match, this indicates that there were no errors during the transmission.

```

- 1d
Write Address : 00000000
Ready..Success
Rewriting:0X0006
Size:0X000094A1  Checksum:91-91
-

```

If you press the mode change button, the downloaded program will be executed.

**Memory Dump** The CM-2+ unit not only has 128 Kbytes of flash memory but also 4 Kbytes of RAM and 4 Kbytes EEPROM. There is a function in the CM-2+ boot loader where you can check the contents in these memory spaces. Type in the memory type as the command, followed by the address. The following figure is an example.

```

- ram 0
00000000 : B5 09 00 00 02 F4 37 00 00 94 00 00 00 00 03 00 .....7.....
00000010 : 10 00 00 00 00 4D 02 2E 01 2E 0A 00 B5 09 1E 00 .....M.....
00000020 : 00 FB DC F8 00 00 00 00 00 00 18 62 00 00 00 2A .....b...*
00000030 : 04 00 00 07 00 07 01 01 01 00 00 00 00 FF FF 0F .....
00000040 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000050 : 00 00 00 00 00 02 00 00 00 00 00 00 00 4F 09 95 .....0..
00000060 : 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 9D .....
00000070 : 00 F8 FE FF 00 00 00 00 00 00 00 00 20 00 00 00 .....

- eeprom 100
00000100 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
00000110 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
00000120 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
00000130 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
00000140 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
00000150 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
00000160 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
00000170 : FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....

- flash 1e000
0001E000 : 0C 94 48 F9 18 95 18 95 18 95 18 95 18 95 18 95 ..H.....
0001E010 : 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 .....
0001E020 : 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 .....
0001E030 : 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 .....
0001E040 : 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 .....
0001E050 : 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 .....
0001E060 : 18 95 18 95 18 95 18 95 18 95 18 95 18 95 18 95 .....
0001E070 : 18 95 18 95 18 95 18 95 0C 94 2C F1 18 95 18 95 .....

```

## 8-2. USING THE C PROGRAM LANGUAGE

The program for the robot was programmed in C and loaded with the Boot Loader. In order to write such a program, you will need to know how to program in C and you should also have some CPU hardware background. This is beyond the scope of this manual, thus we recommend you refer to other references for such information.

In this section we will learn about the Boot Loader and what part of the memory it is located in. We will also learn how much of the memory a user can use for programming.

### Memory Map

The CM-2+ uses a CPU called the Atmega128. This CPU has 128 Kbytes of flash memory. The CM-2+ divides the flash memory into several sections, as shown in the table below.

Address	Item	Function
0X00000 ~ 0X0BFFF	Bioid program	Location of the program that operates the Bioid
0X0A000 ~ 0X0DFFF	User area	Location of the user made behavior control program
0X0E000 ~ 0X1DFFF	Motion Data	Area for storing motion data for the robot
0X1E000 ~ 0X1FFFF	Boot Loader	Location of the "Boot Loader" program for verifying the download and memory status, etc.

When the power is applied, the “Boot Loader” located at address 0x1E000 executes. The file CM-2+ .hex is loaded on to the 48 Kbyte user area, starting at Address 0X00000. You can see that the executable file of the user created C program has to be loaded at the address of 0.

There is a compiler called AVR-GCC for creating the C program that you can use for free. This will be explained in more detail in chapter 9-3.

**TIP**

Learning how to use C to operate the CM-2+ is learning about microprocessors. Studying robotics and studying the microprocessor are two different things. Starting from using the IN and OUT commands in the microprocessor may not be an efficient way to operate a robot. A robot should be considered as a system, not from the low level of a device or a board. When we make a homepage with a PC, we usually don't use ASM or C directly. Rather, we use a higher level tool to do the job. Similar to this, it would be more appropriate to use a higher level tool to concentrate more on the higher level behavior control of the robot.

### 8-3. Compiling(Compile)

This section talks about how to compile a CM-2+ program. Before going through this section, we recommend studying the AX-12 manual.

#### Selecting a Compiler

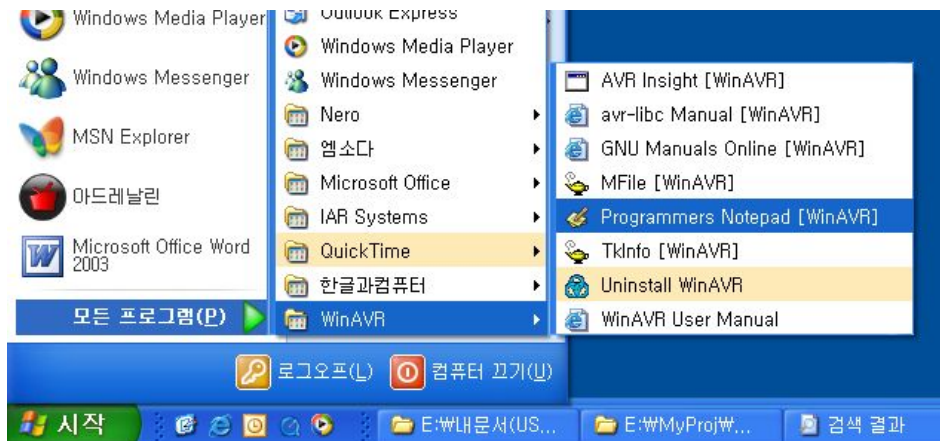
The CPU on the CM-2+ unit is the Atmega128 of the AVR Series from Atmel. There are many different C compilers available for the Atmega128, but their prices are generally very high. On the other hand, a global organization called GNU is distributing their compiler called GCC for free of charge. For this reason, many research labs and institutions are using this compiler instead. The CM-2+ unit also uses the AVR GCC compiler.

## Compiler and Editor

Windows OS users are familiar with compilers that have an editor function built in. For compilers that run on text based OSes such as Linux, they usually have a separate compiler and editor. The GCC has a compiler based on the command line interface and thus does not have a built in editor. Therefore, users have to use a separate editor to develop a program. The AVR-Edit and the WIN-AVR are two editors for GCC that are popular. We will be using the WIN-AVR editor in this tutorial. This editor runs the compiler by internally calling the AVR GCC.

## Installing the Compiler

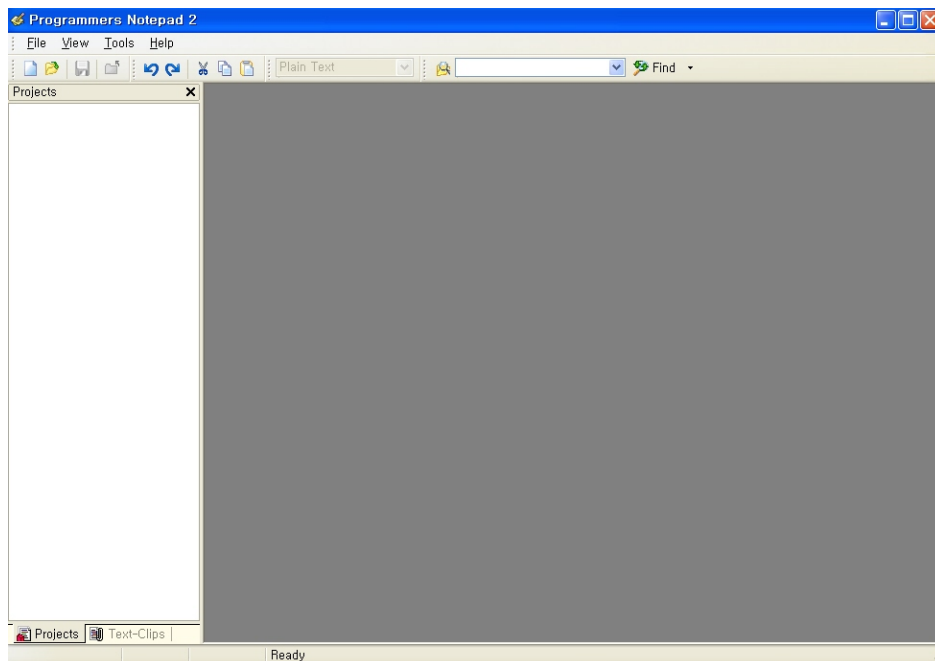
Let's install and run the AVR GCC Editor, Win-AVR. Win-AVR can be downloaded from the Internet and you can find a link from the website [www.robotis.com](http://www.robotis.com). Since Win-AVR already includes the AVR GCC, you only need to install Win-AVR. You can select the following menu after the installation is complete.



## Win AVR

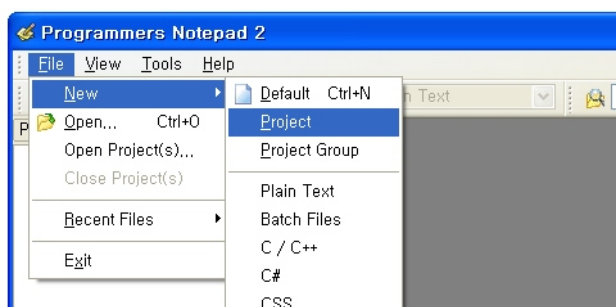
The Win-AVR editor runs the GCC compiler by calling the AVR GCC internally. From the Win-AVR menu, select Programmer's Notepad [Win AVR]. The following screen will appear.



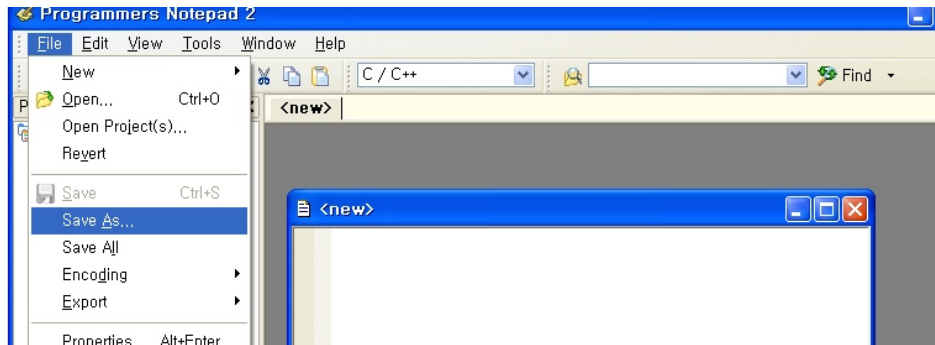


### Project File

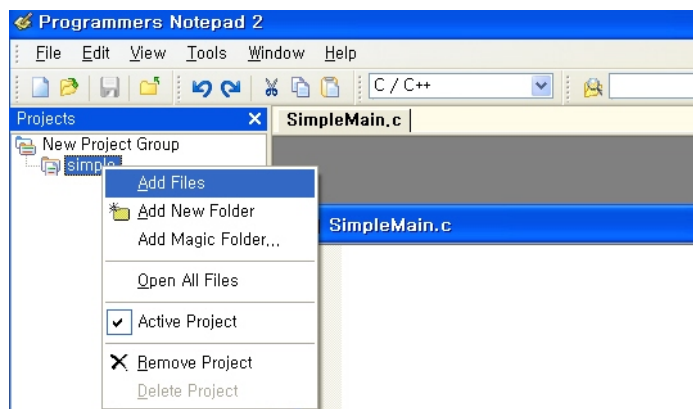
When writing a large program, it is helpful to structure the program by dividing the source file into a number of smaller files by its contents. The Project File is a higher-level file that contains the list of all the source files associated with the program that is being developed and includes all the compile options. Open a new Project File as shown below and give it any name as you wish. Here, the project is named "Simple." Select "Project" from the "New" menu item under the "File" menu.



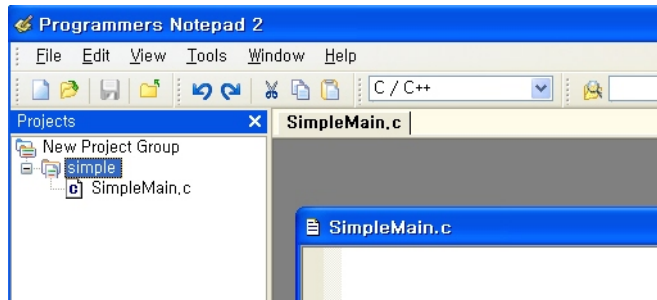
**C Source File** Next, we open the C source code file, which is a lower-level file. Select “C/C++” from the “New” menu item under the “File” menu.



To assign a name, select “Save As...” from the “File” menu and give it any name as you wish. Here, the C source file is named “SimpleMain.c.” Next, the source file named “SimpleMain.c” needs to be added to the project file named “simple”. On the left side of the project window, right click on “simple” then click “Add Files” to select “SimpleMain.c”



Now, a project named “simple” is created, which includes a source file called “SimpleMain.c.”



### Main()

Type the following in the “SimpleMain.c” source code.

```
void main(void)
{
}
```

The above program has no content and is in the form of the most basic structure of a C source code. Now that the source code is completed, the next step is to compile it. To do so, the user has to select the necessary options for compiling it.

### Makefile

The Makefile contains the information for the compile options. Sometimes the project file may contain the information for these options. However, since the Win-VAR does not have an internal compiler, it needs a separate Makefile for the GCC. Just like creating a project, the Makefile needs to be created only once and then modified as needed.

### Location of the Makefile

The Makefile has to be in the same directory (folder location) as the project file and the main source file that contains the Main function. The name of the file has to be Makefile without an extension and cannot be changed. Thus, the files “Makefile,” “Simple.pnproj,” and “SimpleMain.c” have to be in the same folder.

### Editing the Makefile

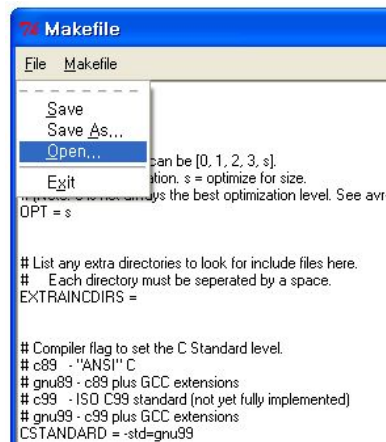
Makefile contains information on opening and compiling the source file, and the name of the executable file. Makefile also contains other information, but the important information that the user needs to deal with are the name of the

resultant file, the name of the source file, and its directory. This concept will become clearer once we go through the following tutorial. First, from the CD that came with the CM-2+ unit, copy "Examples\WC Program\WExample\Wmakefile" to the current working directory folder.

### Running the mfile

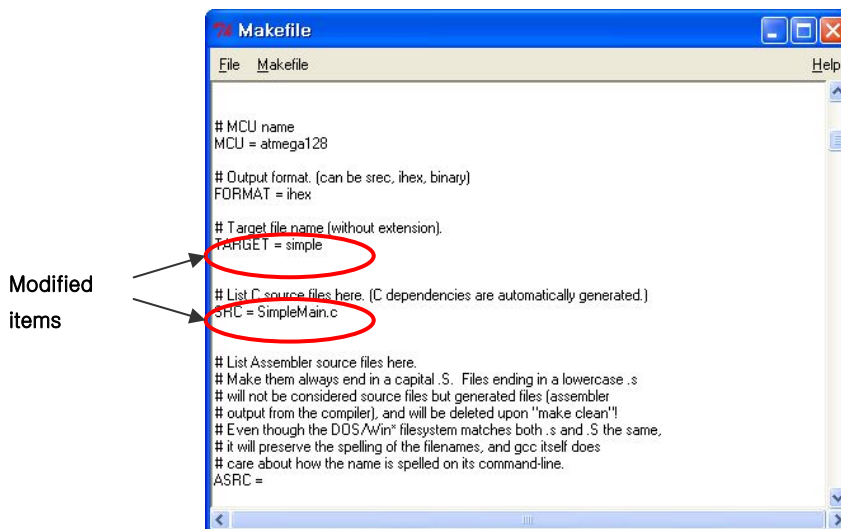
As in the picture shown on the right, run the "mfile" program in Win-AVR. This file only contains the editing function for Makefile.

First, open the file that you want to edit. There are two ways of editing the Makefile; you can directly edit the contents of the Makefile, or you can use the menu to edit it. To edit it using the menu, select the "Makefile" menu on the right to change the options while the "mfile" is running. To directly edit the contents of the Makefile, select the "Enable Editing of Makefile" under the "Makefile" menu to change options by using the keyboard.



### Editing Using the Menu

When a new project is created, two sections have to be modified in the Makefile; one is the main file name section, and the other is the C/C++ Source file(s) section. First set the name of the main file name to "simple." This is used for the file names the compiler creates. Source codes can be added in the C/C++ Source file section. Edit the two sections of the Makefile as shown below.



The Makefile can be edited by using “Notepad” or any text editor.

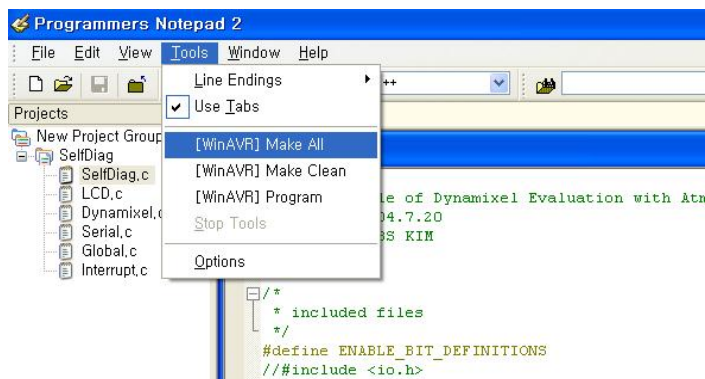
### Summary of Makefile

The concept of Makefile can be tricky for those who use GCC for the first time. The Makefile can be summarized with the following two concepts:

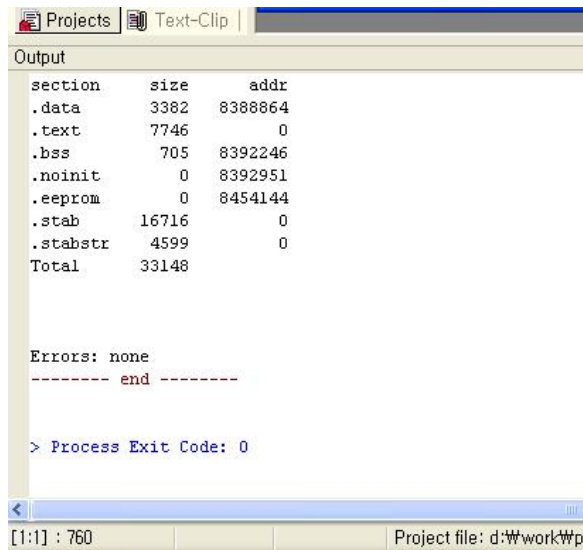
1. The Makefile has to be located in the same folder as the project file and source file. The name of the Makefile cannot be changed.
2. Within the Makefile, the source file section (SRC) and the resultant file section (TARGET) needs to be modified appropriately.

### Executing Compile

Select “Make All” from the “Tools” menu of the Programmers Notepad 2 [WinAVR].



→ output window. If the message will appear.



```
Output
section      size      addr
.data        3382      8388864
.text        7746         0
.bss         705      8392246
.noinit       0      8392951
.eeprom       0      8454144
.stab        16716         0
.stabstr      4599         0
Total        33148

Errors: none
----- end -----

> Process Exit Code: 0

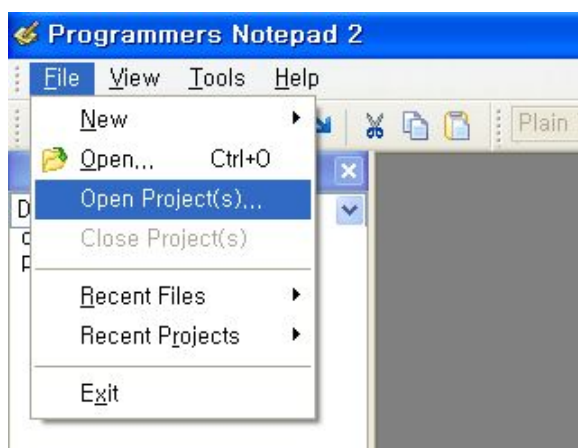
[1:1] : 760      Project file: d:\work\wp
```

### Simple.hex Download

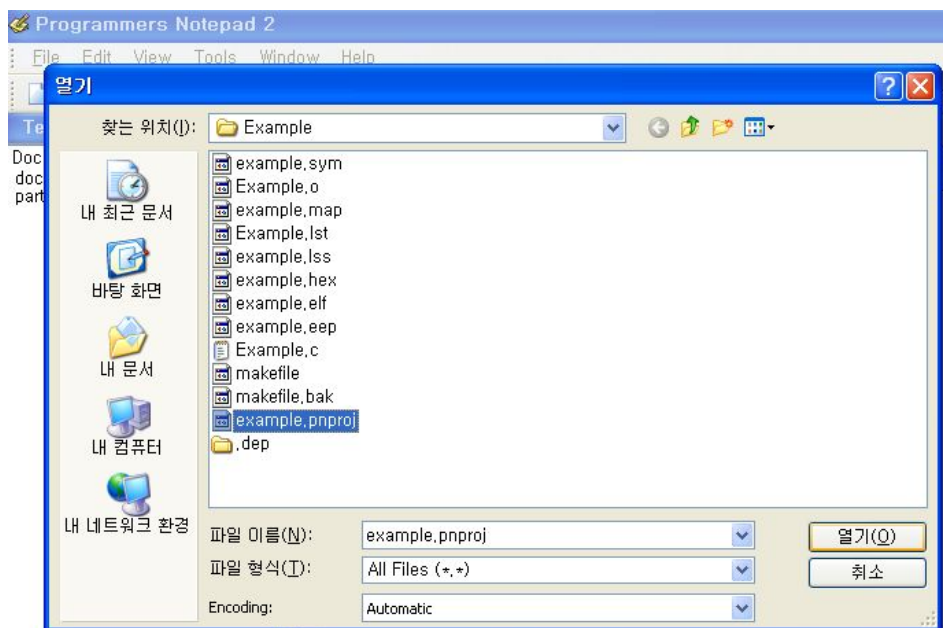
Now let's verify and download the file "simple.hex." Use the Boot Loader to download the program. If you run it, nothing will happen because the file does not contain any information.

## 8-4. Example.c

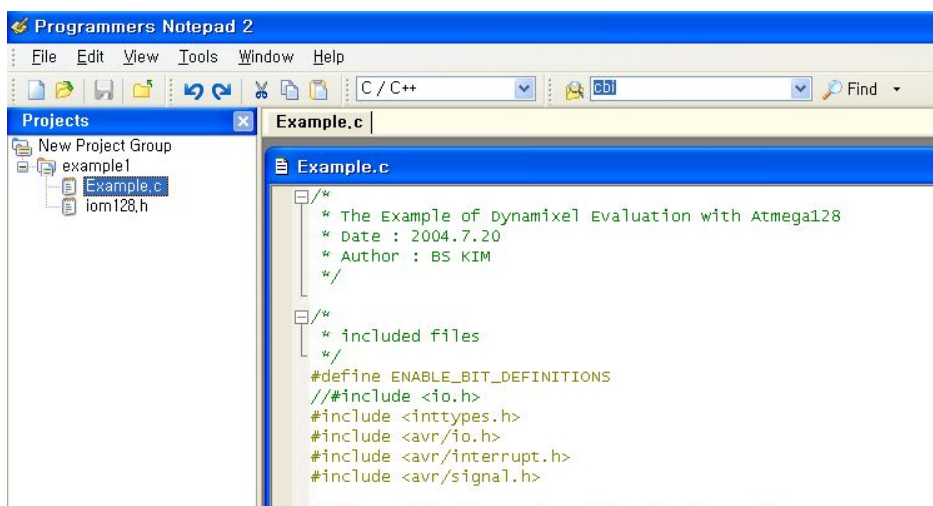
**Example.c** “Example.c” contains various routines for the CM-2+ to directly control the Dynamixel actuators. Using these routines, one can easily develop a program for controlling them. Select “Open Project(s)” from the “File” menu on the MinAVR Programmers Notepad.



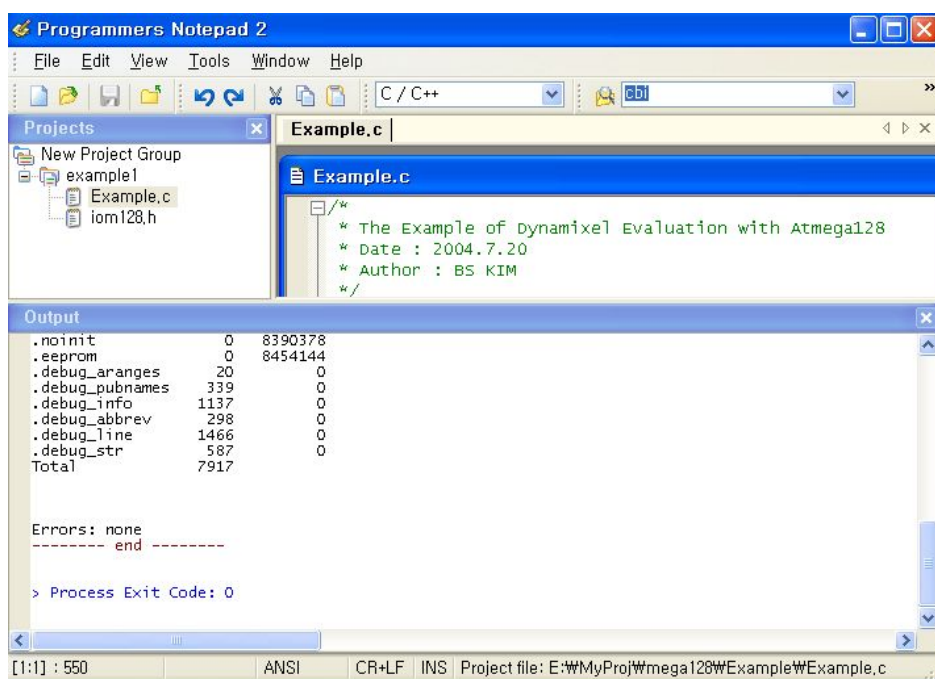
**Project Open** Open the “example.pnproj” project file in the Example folder.



**Files Open** Double-click the “Example.c” on the left and the contents of it appear on the screen.



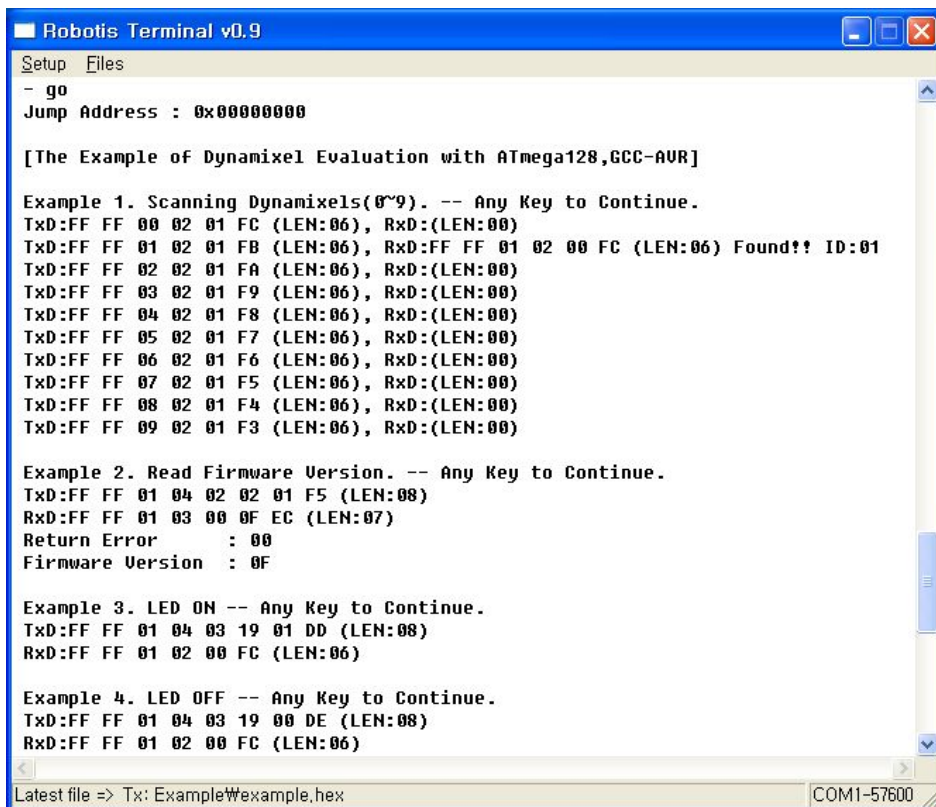
**Compile** Select “Make All” from the “Tools” menu to compile. The output after the compile should look like the following.





## Download

Now, let's use the Robot Terminal to download "example.hex" to the CM-2+ unit. Please refer to Chapter 2 for downloading instructions. Use the "Go" command to execute "example.hex." The screen shot of this is shown below. Pressing a key will make it proceed to the next example.



```

Robotis Terminal v0.9
Setup Files
- go
Jump Address : 0x00000000

[The Example of Dynamixel Evaluation with ATmega128,GCC-AVR]

Example 1. Scanning Dynamixel(0~9). -- Any Key to Continue.
TxD:FF FF 00 02 01 FC (LEN:06), RxD:(LEN:00)
TxD:FF FF 01 02 01 FB (LEN:06), RxD:FF FF 01 02 00 FC (LEN:06) Found?! ID:01
TxD:FF FF 02 02 01 FA (LEN:06), RxD:(LEN:00)
TxD:FF FF 03 02 01 F9 (LEN:06), RxD:(LEN:00)
TxD:FF FF 04 02 01 F8 (LEN:06), RxD:(LEN:00)
TxD:FF FF 05 02 01 F7 (LEN:06), RxD:(LEN:00)
TxD:FF FF 06 02 01 F6 (LEN:06), RxD:(LEN:00)
TxD:FF FF 07 02 01 F5 (LEN:06), RxD:(LEN:00)
TxD:FF FF 08 02 01 F4 (LEN:06), RxD:(LEN:00)
TxD:FF FF 09 02 01 F3 (LEN:06), RxD:(LEN:00)

Example 2. Read Firmware Version. -- Any Key to Continue.
TxD:FF FF 01 04 02 02 01 F5 (LEN:08)
RxD:FF FF 01 03 00 0F EC (LEN:07)
Return Error      : 00
Firmware Version  : 0F

Example 3. LED ON -- Any Key to Continue.
TxD:FF FF 01 04 03 19 01 DD (LEN:08)
RxD:FF FF 01 02 00 FC (LEN:06)

Example 4. LED OFF -- Any Key to Continue.
TxD:FF FF 01 04 03 19 00 DE (LEN:08)
RxD:FF FF 01 02 00 FC (LEN:06)

Latest file => Tx: ExampleWexample.hex
COM1-57600
  
```

Example 1 sends the “Ping” command to the Dynamixel actuators (ID from 0 to 9) and checks if there are replies. The Baud rate for the CM-2+ is set to 57,600 bps. From the results shown here, you can see that one Dynamixel actuator with the ID of 1 is connected to the CM-2+ unit.

Example 2 demonstrates the use of the “Read” command. It reads data from Address 2 of the Control Table of the Dynamixel actuator with the ID of 1. The data from Address 2 is the Firmware version and the results show that it currently has a firmware version of 0x0F. Please refer to the Dynamixel manual for information about the Control Table and for the structure of the packets

Example 3 turns on the LED of a Dynamixel actuator by writing 1 to address 0x19 of the Control Table. All actions for the Dynamixel actuators can be activated in this way by writing data to the corresponding address in the Control Table.

Example 4 turns off the LED of a Dynamixel actuator by writing 0 to address 0x19 of the Control Table.

```

Robotis Terminal v0.9
Setup Files

Example 5. Read Control Table. -- Any Key to Continue.
TxD:FF FF 01 04 02 00 31 C7 (LEN:08)
Rx0:FF FF 01 33 00 74 00 0F 01 22 00 00 00 FF 03 00 55 3C BE FF 03 02 04 04 00
18 00 F6 03 00 00 01 01 20 20 12 00 00 00 FF 03 13 00 00 00 00 00 4A 20 00 00 00
00 20 C4 (LEN:37)
[00]:74 [01]:00 [02]:0F [03]:01 [04]:22 [05]:00 [06]:00 [07]:00 [08]:FF [09]:03
[0A]:00 [0B]:55 [0C]:3C [0D]:BE [0E]:FF [0F]:03 [10]:02 [11]:04 [12]:04 [13]:00
[14]:18 [15]:00 [16]:F6 [17]:03 [18]:00 [19]:00 [1A]:01 [1B]:01 [1C]:20 [1D]:20
[1E]:12 [1F]:00 [20]:00 [21]:00 [22]:FF [23]:03 [24]:13 [25]:00 [26]:00 [27]:00
[28]:00 [29]:00 [2A]:4A [2B]:20 [2C]:00 [2D]:00 [2E]:00 [2F]:00 [30]:20

Example 6. Go 0x200 with Speed 0x100 -- Any Key to Continue.
TxD:FF FF 01 07 03 1E 00 02 00 01 D3 (LEN:0B)
Rx0:FF FF 01 02 00 FC (LEN:06)

Example 7. Go 0x00 with Speed 0x40 -- Any Key to Continue.
TxD:FF FF 01 07 03 1E 00 00 40 00 96 (LEN:0B)
Rx0:FF FF 01 02 00 FC (LEN:06)

Example 8. Go 0x3ff with Speed 0x3ff -- Any Key to Continue.
TxD:FF FF 01 07 03 1E FF 03 FF 03 D2 (LEN:0B)
Rx0:FF FF 01 02 00 FC (LEN:06)

Example 9. Torque OFF -- Any Key to Continue.
TxD:FF FF 01 04 03 18 00 DF (LEN:0B)
Rx0:FF FF 01 02 00 FC (LEN:06)

End. Push reset button for repeat

Latest file => Tx: ExampleWexample.hex COM1-57600

```

Example 5 reads all the data from the Control Table by sending a packet to read data from address 0 to 0x31. The figure above shows the list of these 0x37 packets in the [Address]:Data form.

Example 6 demonstrates the command for moving the output of a Dynamixel actuator to a specified position. This is the most often used command. The Goal Position value of 0x200 (corresponding to the position at 180 degree) is written to address 0x1e of the Control Table. The Goal Speed value of 0x100 is written to address 0x20 of the Control Table as well. Note that both values (Goal Position and Goal Speed) can be written at the same time using only one packet.

Example 7 and Example 8 each demonstrate the command for moving the output of a Dynamixel actuator to a specified position and follow the same method as explain in Example 6.

The last example (Example 9) turns off the torque of the Dynamixel actuator by transmitting a packet to write a 0 to address 0x18 (address for Torque Enable) of the Control Table.

## Example.c

```

/*
 * The Example of Dynamixel Evaluation with Atmega128
 * Date : 2005. 7. 11
 * Author : BS KIM
 */

/*
 * included files
 */
#define ENABLE_BIT_DEFINITIONS
#include <io.h>
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>

#define cbi(REG8, BITNUM) REG8 &= ~(BV(BITNUM))
#define sbi(REG8, BITNUM) REG8 |= BV(BITNUM)

typedef unsigned char byte;
typedef unsigned int word;
#define ON 1
#define OFF 0
#define _ON 0
#define _OFF 1

//--- Control Table Address ---
//EEPROM AREA
#define P_MODEL_NUMBER_L 0
#define P_MODEL_NUMBER_H 1
#define P_VERSION 2
#define P_ID 3
#define P_BAUD_RATE 4
#define P_RETURN_DELAY_TIME 5
#define P_CW_ANGLE_LIMIT_L 6
#define P_CW_ANGLE_LIMIT_H 7
#define P_CCW_ANGLE_LIMIT_L 8
#define P_CCW_ANGLE_LIMIT_H 9
#define P_SYSTEM_DATA2 10
#define P_LIMIT_TEMPERATURE 11
#define P_DOWN_LIMIT_VOLTAGE 12
#define P_UP_LIMIT_VOLTAGE 13
#define P_MAX_TORQUE_L 14
#define P_MAX_TORQUE_H 15
#define P_RETURN_LEVEL 16
#define P_ALARM_LED 17
#define P_ALARM_SHUTDOWN 18
#define P_OPERATING_MODE 19
#define P_DOWN_CALIBRATION_L 20
#define P_DOWN_CALIBRATION_H 21
#define P_UP_CALIBRATION_L 22
#define P_UP_CALIBRATION_H 23

#define P_TORQUE_ENABLE (24)
#define P_LED (25)
#define P_CW_COMPLIANCE_MARGIN (26)
#define P_CCW_COMPLIANCE_MARGIN (27)
#define P_CW_COMPLIANCE_SLOPE (28)
#define P_CCW_COMPLIANCE_SLOPE (29)
#define P_GOAL_POSITION_L (30)
#define P_GOAL_POSITION_H (31)
#define P_GOAL_SPEED_L (32)
#define P_GOAL_SPEED_H (33)
#define P_TORQUE_LIMIT_L (34)
#define P_TORQUE_LIMIT_H (35)
#define P_PRESENT_POSITION_L (36)
#define P_PRESENT_POSITION_H (37)

#define P_PRESENT_SPEED_L (38)
#define P_PRESENT_SPEED_H (39)
#define P_PRESENT_LOAD_L (40)
#define P_PRESENT_LOAD_H (41)
#define P_PRESENT_VOLTAGE (42)
#define P_PRESENT_TEMPERATURE (43)
#define P_REGISTERED_INSTRUCTION (44)
#define P_PAUSE_TIME (45)
#define P_MOVING (46)
#define P_LOCK (47)
#define P_PUNCH_L (48)
#define P_PUNCH_H (49)

//--- Instruction ---
#define INST_PING 0x01
#define INST_READ 0x02
#define INST_WRITE 0x03
#define INST_REG_WRITE 0x04
#define INST_ACTION 0x05
#define INST_RESET 0x06
#define INST_DIGITAL_RESET 0x07
#define INST_SYSTEM_READ 0x0C
#define INST_SYSTEM_WRITE 0x0D
#define INST_SYNC_WRITE 0x83
#define INST_SYNC_REG_WRITE 0x84

#define CLEAR_BUFFER_gbRxBufferReadPointer = gbRxBufferWritePointer
#define DEFAULT_RETURN_PACKET_SIZE 6
#define BROADCASTING_ID 0xfe

#define TxD8 TxD81
#define RxD8 RxD81

//Hardware Dependent Item
#define DEFAULT_BAUD_RATE 34 //57600bps at 16MHz

//////// For CM-2+
#define RS485_TXD PORTE &= ~BV(PE3), PORTE |= BV(PE2)
//_485_DIRECTION = 1
#define RS485_RXD PORTE &= ~BV(PE2), PORTE |= BV(PE3)
//PORT_485_DIRECTION = 0

/*
//////// For CM-2+
#define RS485_TXD PORTE |= BV(PE2); //_485_DIRECTION = 1
#define RS485_RXD PORTE &= ~BV(PE2); //_PORT_485_DIRECTION = 0
*/
//define TXD0_FINISH UCSRA, 6 //This bit is for checking TxD
//Buffer in CPU is empty or not.
//define TXD1_FINISH UCSRA, 6

#define SET_TXD0_FINISH sbi(UCSRA, 6)
#define RESET_TXD0_FINISH cbi(UCSRA, 6)
#define CHECK_TXD0_FINISH bit_is_set(UCSRA, 6)
#define SET_TXD1_FINISH sbi(UCSRA, 6)
#define RESET_TXD1_FINISH cbi(UCSRA, 6)
#define CHECK_TXD1_FINISH bit_is_set(UCSRA, 6)

#define RX_INTERRUPT 0x01
#define TX_INTERRUPT 0x02
#define OVERFLOW_INTERRUPT 0x01
#define SERIAL_PORT0 0
#define SERIAL_PORT1 1
#define BIT_RS485_DIRECTION0 0x08 //Port E
#define BIT_RS485_DIRECTION1 0x04 //Port E

#define BIT_ZIGBEE_RESET PD4 //out : default 1
//PORTD
#define BIT_ENABLE_RXD_LINK_PC PD5 //out : default 1
#define BIT_ENABLE_RXD_LINK_ZIGBEE PD6 //out : default 0

```

```
#define BIT_LINK_PLUGIN          PD7 //in, no pull up

void TxD81(byte bTxData):
void TxD80(byte bTxData):
void TxDString(byte *bData):
void TxD8Hex(byte bSentData):
void TxD32Dec(long lLong):
byte RxD81(void):
void MilliSec(word wDelayTime):
void PortInitialize(void):
void SerialInitialize(byte bPort, byte bBaudrate, byte bInterrupt):
byte TxPacket(byte bID, byte bInstruction, byte bParameterLength):
byte RxPacket(byte bRxLength):
void PrintBuffer(byte *bP, byte bLength):

// --- Gloval Variable Number ---
volatile byte gbpRxInterruptBuffer[256]:
byte gbpParameter[128]:
byte gbpRxBufferReadPointer:
byte gbpRxBuffer[128]:
byte gbpTxBuffer[128]:
volatile byte gbpRxBufferWritePointer:

int main(void)
{
    byte bCount, bID, bTxPacketLength, bRxPacketLength:

    PortInitialize(): //Port In/Out Direction Definition
    RS485_RXD: //Set RS485 Direction to Input State.
    SerialInitialize(SERIAL_PORT0, 1, RX_INTERRUPT): //RS485
        Initializing(RxInterrupt)
    SerialInitialize(SERIAL_PORT1, DEFAULT_BAUD_RATE, 0): //RS232
        Initializing(None Interrupt)

    gbpRxBufferReadPointer = gbpRxBufferWritePointer = 0: //RS485
        RxBuffer Clearing.

    sei(): //Enable Interrupt --- Compiler Function
    TxDString("YrYn [The Example of Dynamixel Evaluation with
        ATmega128, GCC-AVR]");

    //Dynamixel Communication Function Execution Step.
    // Step 1. Parameter Setting (gbpParameter[]). In case of no
        parameter instruction(Ex. INST_PING), this
        step is not needed.
    // Step 2. TxPacket(ID, INSTRUCTION, LengthOfParameter): --Total
        TxPacket Length is returned
    // Step 3. RxPacket(ExpectedReturnPacketLength): -- Real RxPacket
        Length is returned
    // Step 4 PrintBuffer(BufferStartPointer, LengthForPrinting):

    bID = 1:
    TxDString("YrYnYn Example 1. Scanning Dynamixels(0~9). -- Any Key
        to Continue.): RxD8():
    for(bCount = 0; bCount < 0x0A; bCount++)
    {
        bTxPacketLength = TxPacket(bCount, INST_PING, 0):
        bRxPacketLength = RxPacket(255):
        TxDString("YrYn                                TxD:");
                                PrintBuffer(gbpTxBuffer, bTxPacketLength):
        TxDString(",                                RxD:");
                                PrintBuffer(gbpRxBuffer, bRxPacketLength):
        if(bRxPacketLength == DEFAULT_RETURN_PACKET_SIZE)
        {
            TxDString(" Found!! ID:"); TxD8Hex(bCount):
            bID = bCount:
        }
    }

    TxDString("YrYnYn Example 2. Read Firmware Version. -- Any Key to
        Continue.): RxD8():
    gbpParameter[0] = P_VERSION: //Address of Firmware Version
    gbpParameter[1] = 1: //Read Length
    bTxPacketLength = TxPacket(bID, INST_READ, 2):
```

```
bRxPacketLength                                =
                                RxPacket(DEFAULT_RETURN_PACKET_SIZE+gbpParamet
                                er[1]):
    TxDString("YrYn TxD:"); PrintBuffer(gbpTxBuffer, bTxPacketLength):
    TxDString("YrYn RxD:"); PrintBuffer(gbpRxBuffer, bRxPacketLength):
    if(bRxPacketLength == DEFAULT_RETURN_PACKET_SIZE+gbpParameter[1])
    {
        TxDString("YrYn Return Error      :"); TxD8Hex(gbpRxBuffer[4]):
        TxDString("YrYn Firmware Version  :"); TxD8Hex(gbpRxBuffer[5]):
    }

    TxDString("YrYnYn Example 3. LED ON -- Any Key to Continue.):
        RxD8():
    gbpParameter[0] = P_LED: //Address of LED
    gbpParameter[1] = 1: //Writing Data
    bTxPacketLength = TxPacket(bID, INST_WRITE, 2):
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE):
    TxDString("YrYn TxD:"); PrintBuffer(gbpTxBuffer, bTxPacketLength):
    TxDString("YrYn RxD:"); PrintBuffer(gbpRxBuffer, bRxPacketLength):

    TxDString("YrYnYn Example 4. LED OFF -- Any Key to Continue.):
        RxD8():
    gbpParameter[0] = P_LED: //Address of LED
    gbpParameter[1] = 0: //Writing Data
    bTxPacketLength = TxPacket(bID, INST_WRITE, 2):
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE):
    TxDString("YrYn TxD:"); PrintBuffer(gbpTxBuffer, bTxPacketLength):
    TxDString("YrYn RxD:"); PrintBuffer(gbpRxBuffer, bRxPacketLength):

    TxDString("YrYnYn Example 5. Read Control Table. -- Any Key to
        Continue.): RxD8():
    gbpParameter[0] = 0: //Reading Address
    gbpParameter[1] = 49: //Read Length
    bTxPacketLength = TxPacket(bID, INST_READ, 2):
    bRxPacketLength                                =
                                RxPacket(DEFAULT_RETURN_PACKET_SIZE+gbpParamet
                                er[1]):

    TxDString("YrYn TxD:"); PrintBuffer(gbpTxBuffer, bTxPacketLength):
    TxDString("YrYn RxD:"); PrintBuffer(gbpRxBuffer, bRxPacketLength):
    if(bRxPacketLength == DEFAULT_RETURN_PACKET_SIZE+gbpParameter[1])
    {
        TxDString("YrYn");
        for(bCount = 0; bCount < 49; bCount++)
        {
            TxD8(' '); TxD8Hex(bCount): TxDString(":");
            TxD8Hex(gbpRxBuffer[bCount+5]): TxD8(' ');
        }
    }

    TxDString("YrYnYn Example 6. Go 0x200 with Speed 0x100 -- Any Key
        to Continue.): RxD8():
    gbpParameter[0] = P_GOAL_POSITION_L: //Address of Firmware Version
    gbpParameter[1] = 0x00: //Writing Data P_GOAL_POSITION_L
    gbpParameter[2] = 0x02: //Writing Data P_GOAL_POSITION_H
    gbpParameter[3] = 0x00: //Writing Data P_GOAL_SPEED_L
    gbpParameter[4] = 0x01: //Writing Data P_GOAL_SPEED_H
    bTxPacketLength = TxPacket(bID, INST_WRITE, 5):
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE):
    TxDString("YrYn TxD:"); PrintBuffer(gbpTxBuffer, bTxPacketLength):
    TxDString("YrYn RxD:"); PrintBuffer(gbpRxBuffer, bRxPacketLength):

    TxDString("YrYnYn Example 7. Go 0x00 with Speed 0x40 -- Any Key to
        Continue.): RxD8():
    gbpParameter[0] = P_GOAL_POSITION_L: //Address of Firmware Version
    gbpParameter[1] = 0x00: //Writing Data P_GOAL_POSITION_L
    gbpParameter[2] = 0x00: //Writing Data P_GOAL_POSITION_H
    gbpParameter[3] = 0x40: //Writing Data P_GOAL_SPEED_L
    gbpParameter[4] = 0x00: //Writing Data P_GOAL_SPEED_H
    bTxPacketLength = TxPacket(bID, INST_WRITE, 5):
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE):
    TxDString("YrYn TxD:"); PrintBuffer(gbpTxBuffer, bTxPacketLength):
    TxDString("YrYn RxD:"); PrintBuffer(gbpRxBuffer, bRxPacketLength):
```

```

TxDString("\r\n\r\n Example 8. Go 0x3ff with Speed 0x3ff -- Any Key
to Continue."); RxD8();
gbpParameter[0] = P_GOAL_POSITION_L; //Address of Firmware Version
gbpParameter[1] = 0xff; //Writing Data P_GOAL_POSITION_L
gbpParameter[2] = 0x03; //Writing Data P_GOAL_POSITION_H
gbpParameter[3] = 0xff; //Writing Data P_GOAL_SPEED_L
gbpParameter[4] = 0x03; //Writing Data P_GOAL_SPEED_H
bTxPacketLength = TxPacket(bID, INST_WRITE, 5);
bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
TxDString("\r\n\r\n TxID:"); PrintBuffer(gbpTxBuffer, bTxPacketLength);
TxDString("\r\n\r\n RxD:"); PrintBuffer(gbpRxBuffer, bRxPacketLength);

TxDString("\r\n\r\n Example 9. Torque Off -- Any Key to Continue.");
RxD8();
gbpParameter[0] = P_TORQUE_ENABLE; //Address of LED
gbpParameter[1] = 0; //Writing Data
bTxPacketLength = TxPacket(bID, INST_WRITE, 2);
bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
TxDString("\r\n\r\n TxID:"); PrintBuffer(gbpTxBuffer, bTxPacketLength);
TxDString("\r\n\r\n RxD:"); PrintBuffer(gbpRxBuffer, bRxPacketLength);

TxDString("\r\n\r\n End. Push reset button for repeat");
while(1);
}

void PortInitialize(void)
{
  DDRA = DDRB = DDRC = DDRD = DDRE = DDRF = 0; //Set all port to
  input direction first.
  PORTB = PORTC = PORTD = PORTE = PORTF = PORTG = 0x00; //PortData
  initialize to 0
  cbi(SFIO, 2); //All Port Pull Up ready
  DDRE |= (BIT_RS485_DIRECTION0|BIT_RS485_DIRECTION1); //set output
  the bit RS485direction

  DDRD |=
  (BIT_ZIGBEE_RESET|BIT_ENABLE_RXD_LINK_PC|BIT_E
  NABLE_RXD_LINK_ZIGBEE);

  PORTD &= ~_BV(BIT_LINK_PLUGIN); // no pull up
  PORTD |= _BV(BIT_ZIGBEE_RESET);
  PORTD |= _BV(BIT_ENABLE_RXD_LINK_PC);
  PORTD |= _BV(BIT_ENABLE_RXD_LINK_ZIGBEE);
}

/*
TxPacket() send data to RS485.
TxPacket() needs 3 parameter: ID of Dynamixel, Instruction byte,
Length of parameters.
TxPacket() return length of Return packet from Dynamixel.
*/
byte TxPacket(byte bID, byte bInstruction, byte bParameterLength)
{
  byte bCount, bChecksum, bPacketLength;

  gbpTxBuffer[0] = 0xff;
  gbpTxBuffer[1] = 0xff;
  gbpTxBuffer[2] = bID;
  gbpTxBuffer[3] = bParameterLength+2;
  //Length(Paramter, Instruction, Checksum)
  gbpTxBuffer[4] = bInstruction;
  for(bCount = 0; bCount < bParameterLength; bCount++)
  {
    gbpTxBuffer[bCount+5] = gbpParameter[bCount];
  }
  bChecksum = 0;
  bPacketLength = bParameterLength+4+2;
  for(bCount = 2; bCount < bPacketLength-1; bCount++) //except
  0xff,checksum
  {
    bChecksum += gbpTxBuffer[bCount];
  }
  gbpTxBuffer[bCount] = ~bChecksum; //Writing Checksum with Bit
  Inversion

```

```

RS485_TXD:
for(bCount = 0; bCount < bPacketLength; bCount++)
{
  sbi(UCSROA, 6); //SET_TXD0_FINISH;
  TxID0(gbpTxBuffer[bCount]);
}
while(!CHECK_TXD0_FINISH); //Wait until TXD Shift register empty
RS485_RXD:
return(bPacketLength);
}

/*
RxPacket() read data from buffer.
RxPacket() need a Parameter: Total length of Return Packet.
RxPacket() return Length of Return Packet.
*/

byte RxPacket(byte bRxPacketLength)
{
#define RX_TIMEOUT_COUNT2 3000L
#define RX_TIMEOUT_COUNT1 (RX_TIMEOUT_COUNT2*10L)
  unsigned long ulCounter;
  byte bCount, bLength, bChecksum;
  byte bTimeout;

  bTimeout = 0;
  for(bCount = 0; bCount < bRxPacketLength; bCount++)
  {
    ulCounter = 0;
    while(gbRxBufferReadPointer == gbRxBufferWritePointer)
    {
      if(ulCounter++ > RX_TIMEOUT_COUNT1)
      {
        bTimeout = 1;
        break;
      }
    }
    if(bTimeout) break;
    gbpRxBuffer[bCount] = gbRxInterruptBuffer[gbRxBufferReadPointer++];
  }
  bLength = bCount;
  bChecksum = 0;

  if(gbpTxBuffer[2] != BROADCASTING_ID)
  {
    if(bTimeout && bRxPacketLength != 255)
    {
      TxDString("\r\n\r\n [Error:RxD Timeout]");
      CLEAR_BUFFER;
    }

    if(bLength > 3) //checking is available.
    {
      if(gbpRxBuffer[0] != 0xff || gbpRxBuffer[1] != 0xff )
      {
        TxDString("\r\n\r\n [Error:Wrong Header]");
        CLEAR_BUFFER;
        return 0;
      }
      if(gbpRxBuffer[2] != gbpTxBuffer[2] )
      {
        TxDString("\r\n\r\n [Error:TxID != RxID]");
        CLEAR_BUFFER;
        return 0;
      }
      if(gbpRxBuffer[3] != bLength-4)
      {
        TxDString("\r\n\r\n [Error:Wrong Length]");
        CLEAR_BUFFER;
        return 0;
      }
      for(bCount = 2; bCount < bLength; bCount++) bChecksum +=

```

```

        gbpRxBuffer[bCount]:
    if (bChecksum != 0xFF)
    {
        TxDString("YrYn [Error:Wrong CheckSum]");
        CLEAR_BUFFER;
        return 0;
    }
}
return bLength;
}

/*
PrintBuffer() print data in Hex code.
PrintBuffer() needs two parameter: name of Pointer(gbpTxBuffer,
gbpRxBuffer)
*/
void PrintBuffer(byte *bpPrintBuffer, byte bLength)
{
    byte bCount;
    for (bCount = 0; bCount < bLength; bCount++)
    {
        TxD8Hex(bpPrintBuffer[bCount]);
        TxD8(' ');
    }
    TxDString("(LEN:"); TxD8Hex(bLength); TxD8(')');
}

/*
Print value of Baud Rate.
*/
void PrintBaudrate(void)
{
    TxDString("YrYn
        RS232:"); TxD32Dec((16000000L/8L)/((long)UBRR1L
        +1L)); TxDString(" BPS.");
    TxDString(" RS485:"); TxD32Dec((16000000L/8L)/((long)UBRR0L+1L));
    TxDString(" BPS.");
}

/*Hardware Dependent Item*/
#define TXD1_READY          bit_is_set(UCSR1A, 5)
                        //(UCSR1A_Bit5)
#define TXD1_DATA          (UDR1)
#define RXD1_READY         bit_is_set(UCSR1A, 7)
#define RXD1_DATA          (UDR1)

#define TXD0_READY         bit_is_set(UCSR0A, 5)
#define TXD0_DATA          (UDR0)
#define RXD0_READY         bit_is_set(UCSR0A, 7)
#define RXD0_DATA          (UDR0)

/*
SerialInitialize() set Serial Port to initial state.
Vide Mega128 Data sheet about Setting bit of register.
SerialInitialize() needs port, Baud rate, Interrupt value.
*/
void SerialInitialize(byte bPort, byte bBaudrate, byte bInterrupt)
{
    if (bPort == SERIAL_PORT0)
    {
        UBRR0H = 0; UBRR0L = bBaudrate;
        UCSROA = 0x02; UCSROB = 0x18;
        if (bInterrupt & RX_INTERRUPT) sbi(UCSROB, 7); // RxD interrupt
            enable
        UCSROC = 0x06; UDRO = 0xFF;
        sbi(UCSROA, 6); //SET_TXD0_FINISH: // Note. set 1, then 0 is read
    }
    else if (bPort == SERIAL_PORT1)
    {
        UBRR1H = 0; UBRR1L = bBaudrate;

```

```

        UCSR1A = 0x02; UCSR1B = 0x18;
        if (bInterrupt & RX_INTERRUPT) sbi(UCSR1B, 7); // RxD interrupt
            enable
        UCSR1C = 0x06; UDR1 = 0xFF;
        sbi(UCSR1A, 6); //SET_TXD1_FINISH: // Note. set 1, then 0 is read
    }
}

/*
TxD8Hex() print data seprately.
ex> 0x1a -> '1' 'a'.
*/
void TxD8Hex(byte bSentData)
{
    byte bTmp;

    bTmp = (byte) (bSentData >> 4) & 0x0f + (byte) '0';
    if (bTmp > '9') bTmp += 7;
    TxD8(bTmp);
    bTmp = (byte) (bSentData & 0x0f) + (byte) '0';
    if (bTmp > '9') bTmp += 7;
    TxD8(bTmp);
}

/*
TxD80() send data to USART 0.
*/
void TxD80(byte bTxdData)
{
    while (!TXD0_READY);
    TXD0_DATA = bTxdData;
}

/*
TXD81() send data to USART 1.
*/
void TxD81(byte bTxdData)
{
    while (!TXD1_READY);
    TXD1_DATA = bTxdData;
}

/*
TXD32Dex() change data to decimal number system
*/
void TxD32Dec(long lLong)
{
    byte bCount, bPrinted;
    long lTmp, lDigit;
    bPrinted = 0;
    if (lLong < 0)
    {
        lLong = -lLong;
        TxD8('-');
    }
    lDigit = 1000000000L;
    for (bCount = 0; bCount < 9; bCount++)
    {
        lTmp = (byte) (lLong/lDigit);
        if (lTmp)
        {
            TxD8(((byte) lTmp) + '0');
            bPrinted = 1;
        }
        else if (bPrinted) TxD8(((byte) lTmp) + '0');
        lLong -= ((long) lTmp) * lDigit;
        lDigit = lDigit/10;
    }
    lTmp = (byte) (lLong/lDigit);
    /*if (lTmp)*/ TxD8(((byte) lTmp) + '0');
}

/*
TxDString() prints data in ACSII code.

```

```
*/
void TxDString(byte *bData)
{
    while(*bData)
    {
        TxD8(*bData++);
    }
}

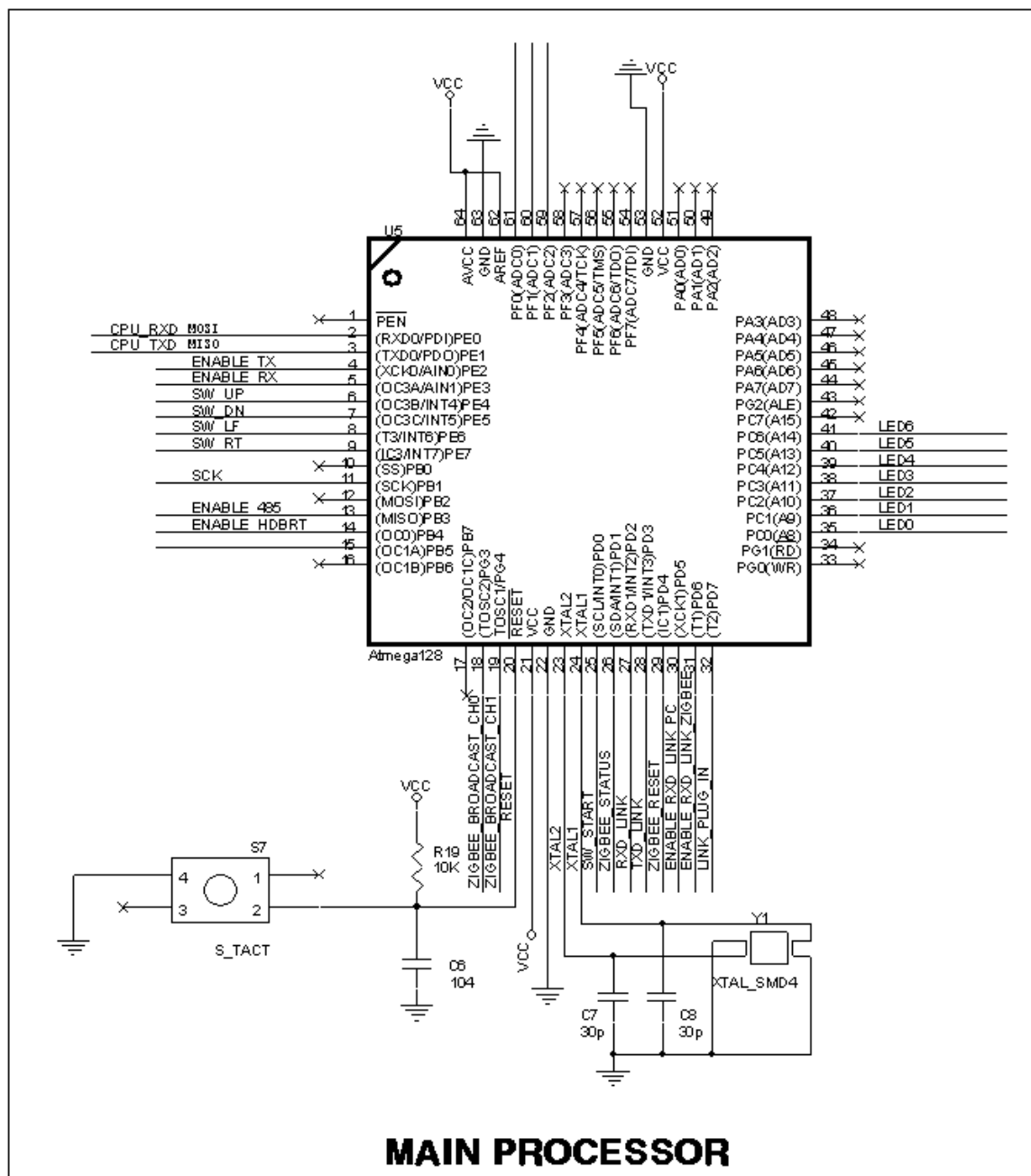
/*
RxData() read data from UART1.
RxData() return Read data.
*/
byte RxData(void)
```

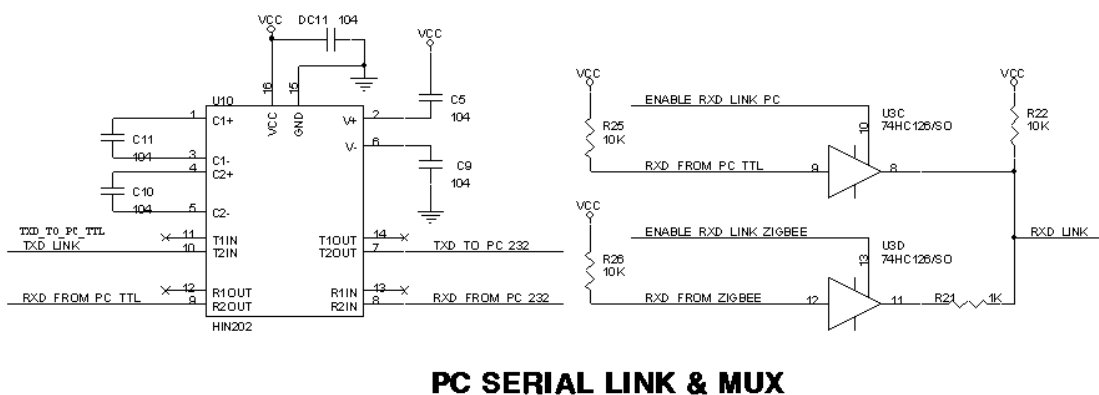
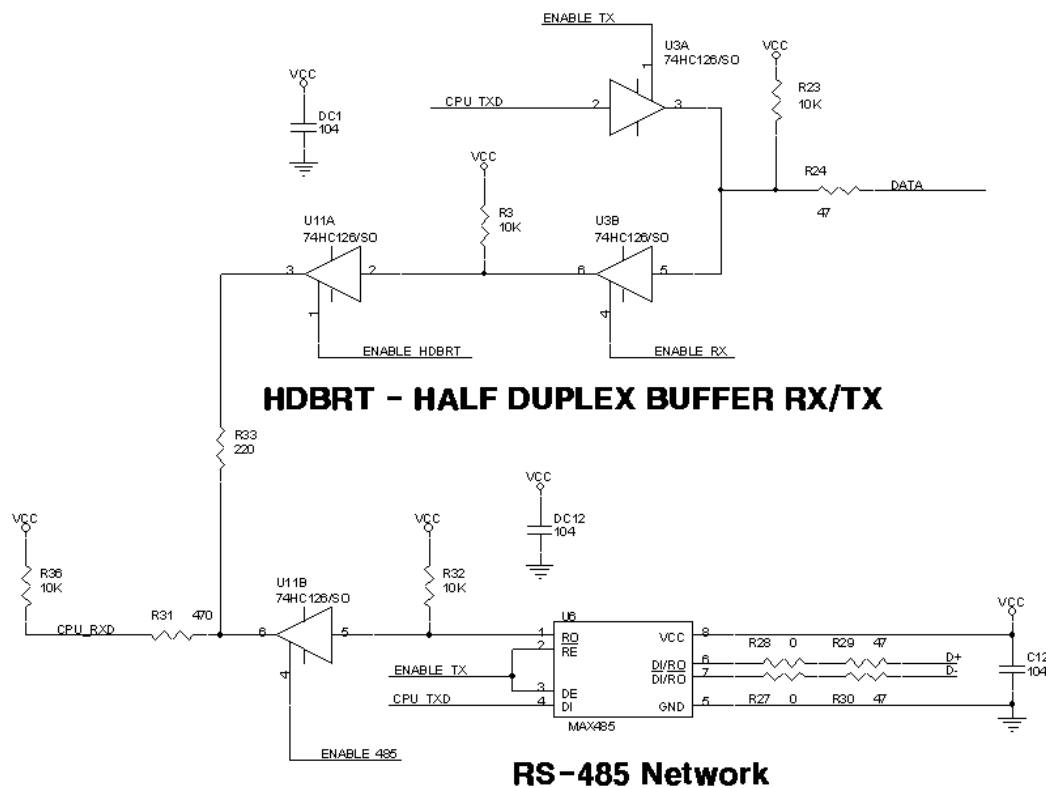
```
{
    while(!RXD1_READY);
    return(RXD1_DATA);
}

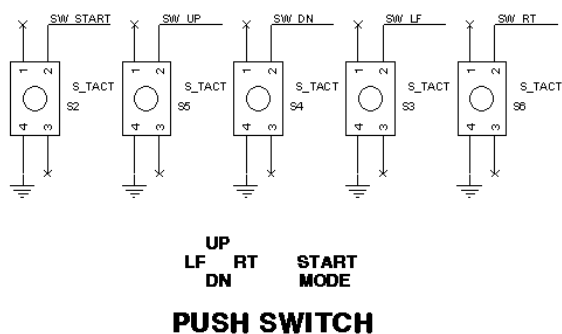
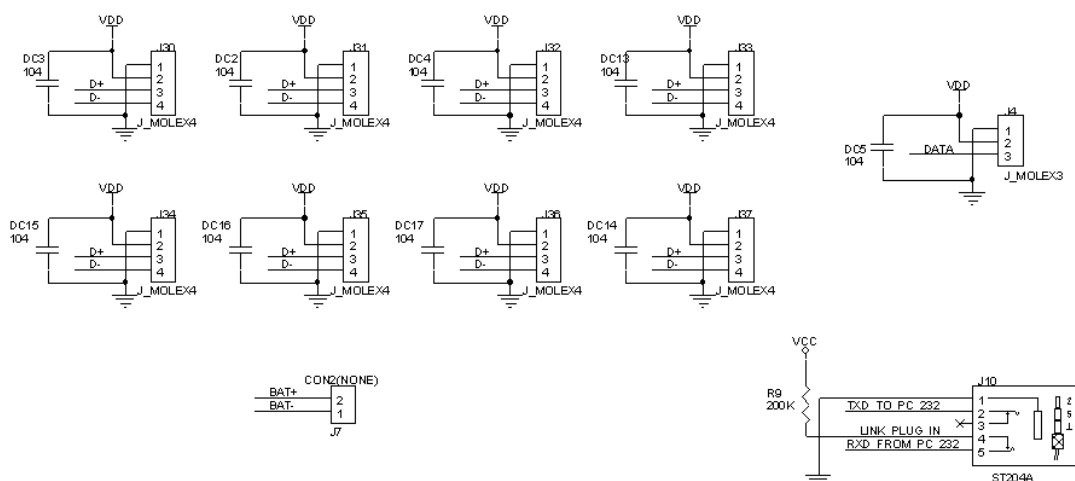
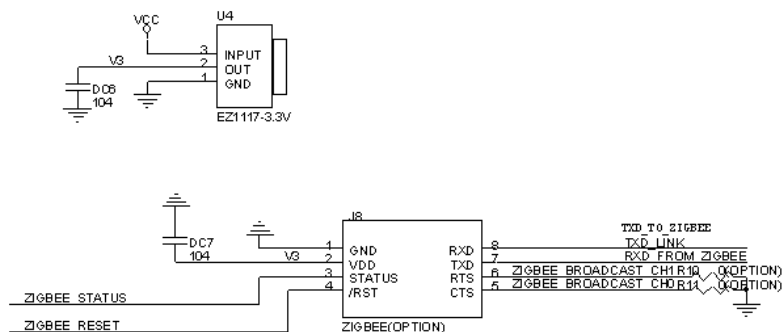
/*
SIGNAL() UART0 Rx Interrupt - write data to buffer
*/
SIGNAL (SIG_UART0_RECV)
{
    gbpRxInterruptBuffer[(gbRxBufferWritePointer++)] = RXD0_DATA;
}
```



## 8-5. Circuit Diagram







## 8-6. Understanding CPU Port

Name	Usage	DIR	Initial Value	Comment
PB1	SCK	IN	0	for ISP Programming
PB3	ENABLE_485	OUT	1	UART0 Select (485) - Note (1)
PB4	ENABLE_HDBRT	OUT	0	UART0 Select (HDBRT) - Note (1)
PB5	Reserved	OUT	1	- Note (2)
PC0	LED0	OUT	1	Power LED
PC1	LED1	OUT	0	TXD LED
PC2	LED2	OUT	0	RXD LED
PC3	LED3	OUT	0	AUX LED
PC4	LED4	OUT	0	MANAGE LED
PC5	LED5	OUT	0	PROGRAM LED
PC6	LED6	OUT	0	PLAY LED
PD0	SW_START	IN	1	START Button
PD1	ZIGBEE_STATUS	IN	0	ZIGBEE Module Status LED Pin
PD2	RXD_LINK	IN	1	UART1 Rxd (PC/ZIGBEE)
PD3	TXD_LINK	OUT	1	UART1 Txd (PC/ZIGBEE)
PD4	ZIGBEE_RESET	OUT	1	ZIGBEE Module Reset Pin
PD5	ENABLE_RXD_LINK_PC	OUT	1	UART1 Select (PC) - Note (1)
PD6	ENABLE_RXD_LINK_ZIGBEE	OUT	0	UART1 Select (ZIGBEE) - Note (1)
PD7	LINK_PLUG_IN	IN	0	PC Serial Link (1: Linked, 0: Not Linked)
PE0	CPU_RXD	IN	1	UART0 Rxd (485/HDBRT)
PE1	CPU_TXD	OUT	1	UART0 Txd (485/HDBRT)
PE2	ENABLE_TX	OUT	0	RS-485/HDBRT Tx Direction Control - Note

		T OU T		(1)
PE3	ENABLE_RX		1	HDBRT Rx Direction Control – Note (1)
PE4	SW_UP	IN	1	UP Button
PE5	SW_DN	IN	1	Down Buton
PE6	SW_LF	IN	1	LEFT Button
PE7	SW_RT	IN	1	RIGHT Button
PF0	Reserved	IN	0	– Note (2)
PF1	Reserved	IN	0	– Note (2)
PF2	Reserved	IN	0	– Note (2)
PG3	ZIGBEE_BROADCAST_CH0	IN	0	ZIGBEE Module Channel Select 0
PG4	ZIGBEE_BROADCAST_CH1	IN	0	ZIGBEE Module Channel Select 1

% Precautions on using CM-2+ CPU port

Note(1)	As it is for selecting connection, both ports cannot be 1.
Note(2)	It must maintain initial value.

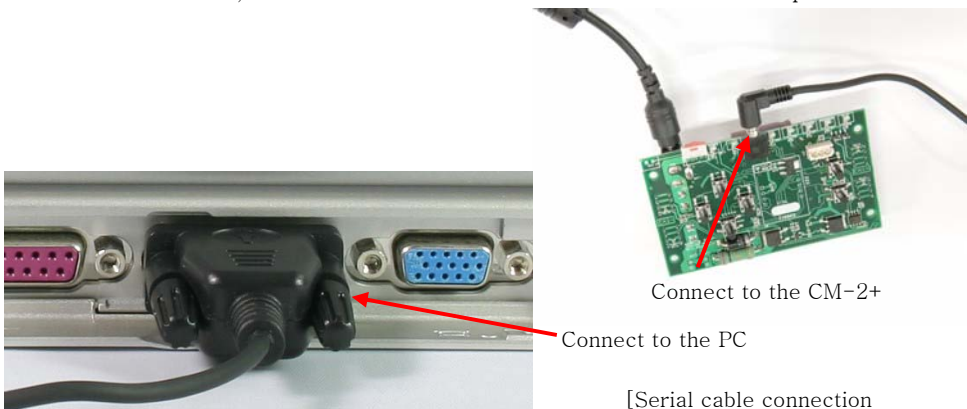
## 9. CM-2+ Program Update

This chapter is about the robot program update. We are going to introduce a way to maintain Bioloid in latest version by showing how to update the firmware for the CM-2+ , the main controller, and the Dynamixel motor. We recommend that you visit the Robotis site, [www.robotis.com](http://www.robotis.com), and download the latest version.

### 9-1. CM-2+ Program Update

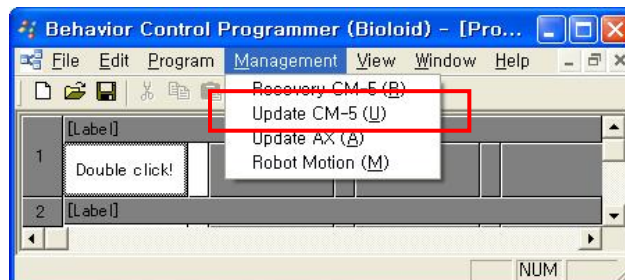
The CM-2+ program is updated through the behavior control programmer. Follow the process below to update to the latest version.

[STEP 1] As shown below, connect to the PC and CM-2+ and turn on the power.

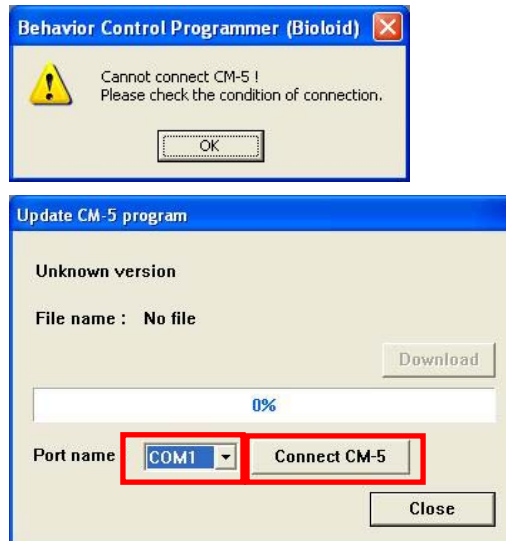


[STEP 2] Run the behavior control programmer.

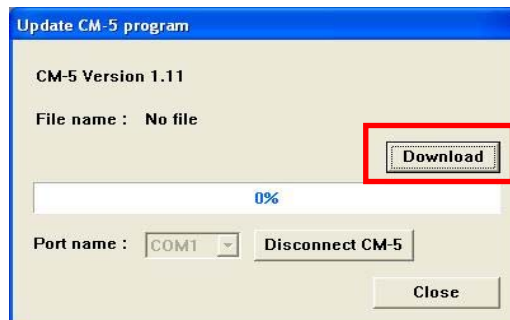
[STEP 3] Select "Management" -> "CM-5 Update" in the menu.



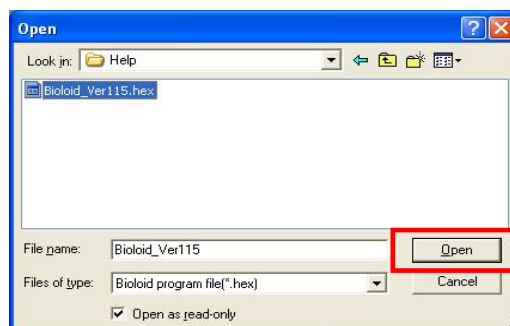
- [STEP 4] When a message that says “The CM-5 is not found” appears, correctly set the “communication port” and click the “Connecting CM-5” button.



- [STEP 5] When the connection is made, click the ‘Download’ button.

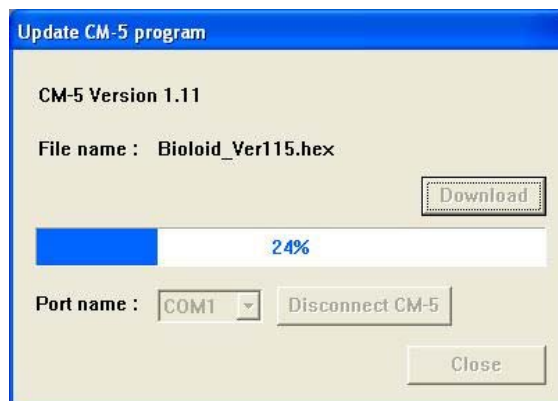


- [STEP 6] Select the CM-2+ program file.  
You can download data files needed for update from our ROBOTIS website ([www.robotis.com](http://www.robotis.com)). It is recommended to always update the program with the latest file.

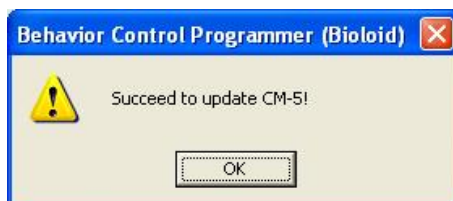


**[STEP 7]**

While the update proceeds, the proceeding state is displayed.  
Be careful that power is not shut off while the update proceeds.

**[STEP 8]**

When the update is completed, a dialogue box saying that the update is successfully completed will be displayed. When an error occurs, retry from the first step.





## 9-2. Dynamixel Program Update

Programs of the Dynamixels are upgraded through the behavior control program. Proceed with the update according to the following procedure.

The upgrade of the AX-12+ is a newly added function from the version 1.26 of the behavior control program. Therefore, when you use the program of any version before 1.26, you should download and install the latest version of Boiloid software from our ROBOTIS website([www.robotis.com](http://www.robotis.com)) to proceed with the upgrade. You can verify the version of the behavior control program by selecting “Help -> Behavior Control Programmer Information”.

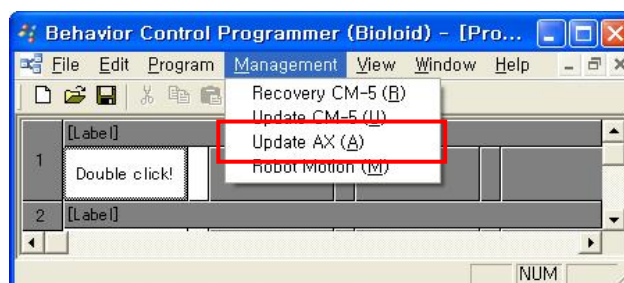


In addition, the upgrade is possible only when the version of the CM-2+ program is more than 1.13. Therefore, if you use the program of a version before 1.13, download the latest CM-2+ program from our ROBOTIS website ([www.robotis.com](http://www.robotis.com)) and proceed with the upgrade of the CM-2+ first.

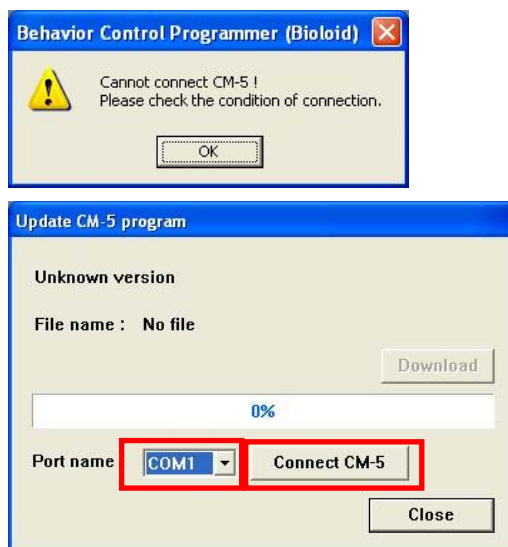
[STEP 1] Connect the CM-2+ and the PC In the same way as the CM-2+ upgrade and turn the CM-2+ power switch on. At this moment, the AX-12+ to be updated should be connected to the CM-2+ . The ID of the AX-12+ to be updated should be a value from 1 to 19. If there is an ID of the AX-12+ out of this range, modify the ID first. When the AX-12+ s with duplicated IDs are connected to the CM-2+ , the update is not properly proceeded.

[STEP 2] Run the behavior control programmer.

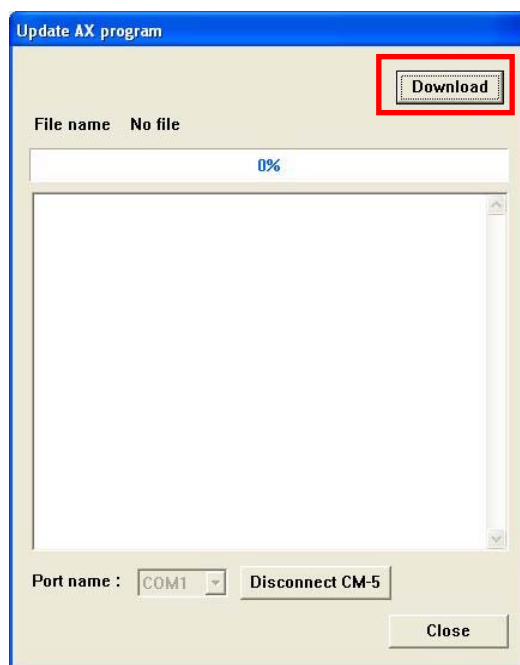
[STEP 3] Select “Management” -> “AX Update” in the menu as the following picture.



- [STEP 4] When a message saying that “CM-5 is not found” is displayed, correctly set the “communication port” and click the “connecting CM-5” button.



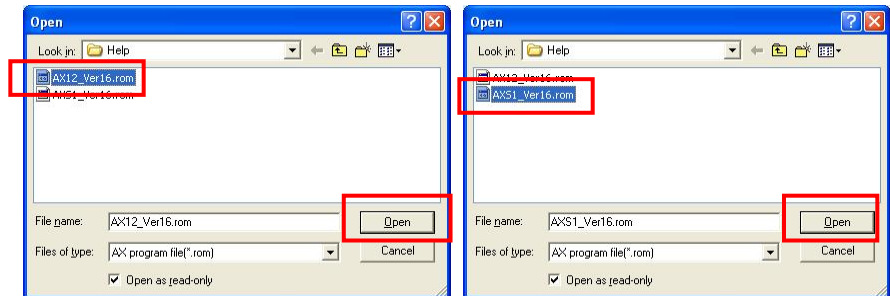
- [STEP 5] When the connection is made, click the ‘Download’ button.



**[STEP 6]**

Select the program file of the AX-12+.

You can download data files needed for update from our ROBOTIS website ([www.robotis.com](http://www.robotis.com)). It is recommended to always update the program with the latest file.

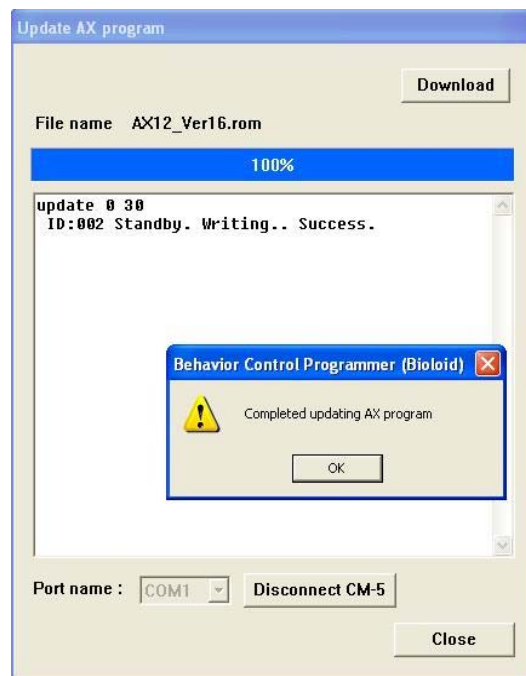


Dynamixel

AX-S1

**[STEP 7]**

The procedure of the program update is displayed in the output window. Be careful that power is not shut off while the update proceeds.

**[STEP 8]**

When a message saying that 'the update is completed' is displayed, click the "Close" button to finish the AX-12+ program update.

## APPENDIX

CM-2+ board diagram

